AD-A103 283 CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF PSYCHOLOGY F/G 5/10
ACQUISITION OF COGNITIVE SKILL.(U)
AUG 81 J R ANDERSON N00014-81-C-0335
UNCLASSIFIED TR-81-1 NL

1 of 2
AD A
103283

LEVEL ✓

(5)

AD A103283

(1) # Acquisition of Cognitive Skill.

(10) John R. Anderson
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

(14) TR-81-1

(9) Technical rept.

(11) 3 Aug 81 (12) 97

Approved for public release; distribution unlimited.
Reproduction in whole or part is permitted for any purpose of the United States Government.

(15) N00014-81-C-0335
NSF-IST80-15357

This research was sponsored by the Personnel and Training Research Programs, Psychological Sciences Division, Office of Naval Research, under Contract No. N00014-81-C-0335, Contract Authority Identification Number, NR 157-465 and grant IST-80-15357 from the National Science Foundation. My ability to put together this theory has depended critically on input from my collaborators over the past few years--Charles Beasley, Jim Greeno, Paul Kline, Pat Langley, and David Neves. This is not to suggest that any of the above would endorse all of the ideas in this paper. I would like to thank those who have provided me with valuable advice and feedback on the paper--Renee Elio, Jill Larkin, Clayton Lewis, Miriam Schustack, and especially Lynne Reder. Correspondence concerning the manuscript should be sent to John Anderson, Department of Psychology, Carnegie-Mellon University, Pittsburgh, Pa. 15213.

384826

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report 81-1	2. GOVT ACCESSION NO. AD-A103 283	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Acquisition of Cognitive Skill		5. TYPE OF REPORT & PERIOD COVERED Technical Report
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) John R. Anderson		8. CONTRACT OR GRANT NUMBER(s) N00014-81-C-0335
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Psychology Carnegie-Mellon University Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NR 157-465
11. CONTROLLING OFFICE NAME AND ADDRESS Personnel and Training Research Programs Office of Naval Research (Code 458) Arlington, VA 22217		12. REPORT DATE August 3, 1981
		13. NUMBER OF PAGES 96
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Geometry	Problem solving	Automatization
Mathematics education	Representation	Declarative knowledge
Skill acquisition	Proceduralization	Procedural knowledge
Learning	Analogy	Practice effects
Production systems	Discrimination	Tuning

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A framework for skill acquisition is proposed in which there are two major stages in the development of a cognitive skill--a declarative stage in which facts about the skill domain are interpreted and a procedural stage in which the domain knowledge is embodied directly in procedures for performing the skill. This general framework has been instantiated in the ACT system in which facts are encoded in a propositional network and procedures are encoded as productions. Two types of interpretive procedures are described for converting facts in the declarative stage into behavior--general problem-solving procedures and analogy-

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract (Continued)

forming procedures. Knowledge compilation is the process by which the skill transits from the declarative stage to the procedural stage. It consists of the subprocesses of composition which collapses sequences of productions into single productions and proceduralization which embeds factual knowledge into productions. Once proceduralized, further learning processes operate on the skill to make the production more selective in their range of applications. These learning processes include generalization, discrimination, and strengthening of productions. Comparisons are made to similar concepts from past learning theories. It is discussed how these learning mechanisms apply to produce the power law speed-up in processing time with practice. Much of the evidence for this theory of skill acquisition comes from work on acquisition of proof skills in geometry but other evidence is drawn from the literature on automatization, language acquisition, and category formation.

19. Key Words (Continued)

Interpretive procedures	Proof skills	Category formation
Knowledge compilation	Language acquisition	Power law
Strengthening		

TABLE OF CONTENTS

Abstract

A framework for skill acquisition is proposed in which there are two major stages in the development of a cognitive skill--a declarative stage in which facts about the skill domain are interpreted and a procedural stage in which the domain knowledge is embodied directly in procedures for performing the skill. This general framework has been instantiated in the ACT system in which facts are encoded in a propositional network and procedures are encoded as productions. Two types of interpretive procedures are described for converting facts in the declarative stage into behavior--general problem-solving procedures and analogy-forming procedures. Knowledge compilation is the process by which the skill transits from the declarative stage to the procedural stage. It consists of the subprocesses of composition which collapses sequences of productions into single productions and proceduralization which embeds factual knowledge into productions. Once proceduralized, further learning processes that operate on the skill to make the productions more selective in their range of applications. These learning processes include generalization, discrimination, and strengthening of productions. Comparisons are made to similar concepts from past learning theories. It is discussed how these learning mechanisms apply to produce the power law speed-up in processing time with practice. Much of the evidence for this theory of skill acquisition comes from work on acquisition of proof skills in geometry but other evidence is drawn from the literature on automatization, language acquisition, and category formation.

Introduction

It requires at least a hundred hours of learning and practice to acquire any significant cognitive skill to a reasonable degree of proficiency. For instance, after 100 hours a student learning to program has achieved only a very modest facility in the skill. Learning one's primary language takes tens of thousands of hours. The psychology of human learning has been very thin in ideas about what happens to skills under the impact of this amount of learning--and for obvious reasons. This paper presents a theory about the changes in the nature of a skill over such large time scales and about the basic learning processes that are responsible.

Fitts (1964) considered the process of skill acquisition to fall into three stages of development. The first stage, called the *cognitive stage*, involves an initial encoding of the skill into a form sufficient to permit the learner to generate the desired behavior to at least some crude approximation. In this stage it is common to observe verbal mediation in which the learner rehearses information required for the execution of the skill. The second stage, called the *associative stage*, involves the "smoothing out" of the skill performance. Errors in the initial understanding of the skill are gradually detected and eliminated. Concomitant with this is the drop out of verbal mediation. The third stage, the *autonomous stage*, is one of gradual continued improvement in the performance of the skill. The improvements in this stage often continue indefinitely. While these general observations about the course of skill development seem true for a wide range of skills, they have defied systematic theoretical analysis.

The theory to be presented in this paper is in keeping with these general observations of Fitts and provides an explanation of the phenomena associated with his three stages. In fact, the three major sections of this paper correspond to these three stages. In the first stage, the learner receives instruction and information about a skill. The instruction is encoded as a set of facts about the skill. These facts can be used by general *interpretive procedures* to generate behavior. This initial stage of skill corresponds to Fitts' cognitive stage. In the paper this will be referred to as the *declarative stage*. Verbal mediation is frequently observed because the facts have to be rehearsed in working memory to keep them available for the interpretive procedures.

According to the theory to be presented here, Fitts' second stage is really a transition between the declarative stage and a later stage. With practice the knowledge is converted into a procedural form in which it is directly applied without the intercession of other interpretive procedures. The gradual process by which the knowledge is converted from declarative to procedural form is called *knowledge compilation*. Fitts' associative stage corresponds to the period over which knowledge compilation applies.

According to the theory, Fitts' autonomous stage, involves further learning that occurs after the knowledge achieves procedural form. In particular, there is further tuning of the knowledge so that it it will apply more appropriately and there is a gradual process of speed-up. This will be called the *procedural stage* in this paper.

This paper presents a detailed theory about the use and development of knowledge in both the declarative and procedural form and about the transition between these two forms. The theory is based on the ACT production system (Anderson, 1976) in which the distinction between procedural and declarative knowledge is fundamental. Procedural knowledge is represented as productions whereas declarative knowledge is represented as a propositional network. Before describing the theory of skill acquisition it will be necessary to specify some of the basic operating principles of the ACT production system.

The ACT Production System

The ACT production system consists of a set of productions which can operate on facts in the declarative data base. Each production has the form of a primitive rule that specifies a cognitive contingency--that is to say, a production specifies when a cognitive act should take place. The production has a *condition* which specifies the circumstances under which the production can apply and an *action* which specifies what should be done when production applies. The sequence of productions that apply in a task correspond to the cognitive steps taken in performing the task. In the actual computer simulations these production rules have often quite technical syntax, but in this paper I will usually give the rules quite English-like renditions. For current purposes, application of a production can be thought of as a step of cognition. Much of the ACT performance theory is concerned with specifying how productions are selected to apply and much of the ACT learning theory is concerned with how these production rules are acquired.

An Example

To explain some of the basic concepts of the ACT production system, it is useful to have an example set of productions which perform some simple task. Such a set of productions for performing addition is given in Table 1. Figure 1 illustrates the flow of control in that production set among goals. It is easiest to understand such a production system by tracing its application to a problem such as the following:

614
438
<u>683</u>

Production P1 is the first to apply and would set as a subgoal to iterate through the columns. Then production P2 applies and changes the subgoal to adding the digits of the rightmost column. It also sets the running total to 0. Then production P6 applies to set the new subgoal to adding the top digit of the row (4) to the running total. In terms of Figure 1 this sequence of three productions has moved the system down from the top goal of doing the problem to the bottom goal of performing a basic addition operation. The system has the four goals in Figure 1 stacked with attention focused on the bottom goal.

At this point production P10 applies which calculates 4 as the new value of the running total and POPs the goal of adding the digit to the running total. This amounts to removing this goal from the stack and returning

Table 1
A Production System for Performing Addition

```
P1:      IF the goal is to do an addition problem
       THEN the subgoal is to iterate through the columns of the problem

P2:      IF the goal is to iterate through the columns of an addition problem
            and the rightmost column has not been processed
       THEN the subgoal is to iterate through the rows of that rightmost column
            and set the running total to 0

P3:      IF the goal is to iterate through the columns of an addition problem
            and a column has just been processed
            and another column is to the left of this column
       THEN the subgoal is to iterate through the rows of this column to the left
            and set the running total to the carry

P4:      IF the goal is to iterate through the columns of an addition problem
            and the last column has been processed
            and there is a carry
       THEN write out the carry
            and POP the goal

P5:      IF the goal is to iterate through the columns of an addition problem
            and the last column has been processed
            and there is no carry
       THEN POP the goal

P6:      IF the goal is to iterate through the rows of a column
            and the top row has not been processed
       THEN the subgoal is to add the digit of the top row into the running total

P7:      IF the goal is to iterate through the rows of a column
            and a row has just been processed
            and another row is below it
       THEN the subgoal is to add the digit of the lower row to the running total

P8:      IF the goal is to iterate through the rows of a column
            and the last row has been processed
            and the running total is a digit
       THEN write the digit
            and delete the carry
            and mark the column as processed
            and POP the goal
```

```
P9:        IF the goal is to iterate through the rows of a column
              and the last row has been processed
              and the running total is of the form "string + digit"
         THEN write the digit
              and set carry to the string
              and mark the column as processed
              and POP the goal

P10:       IF the goal is to add a digit to a number
              and the number is a digit
              and a sum is the sum of the two digits
         THEN the result is the sum
              and mark the digit as processed
              and POP the goal

P11:       IF the goal is to add a digit to a number
              and the number is of the form string + digit
              and a sum is the sum of the two digits
              and the sum is less than 10
         THEN the result is string + sum
              and mark the digit as processed
              and POP the goal

P12:       IF the goal is to add a digit to a number
              and the number is of the form string + digit
              and a sum is the sum of the two digits
              and the sum is of the form 1 + digit*
              and another number sum* is the sum of 1 plus string
         THEN the result is sum* + digit*
              and mark the digit as processed
              and POP the goal
```

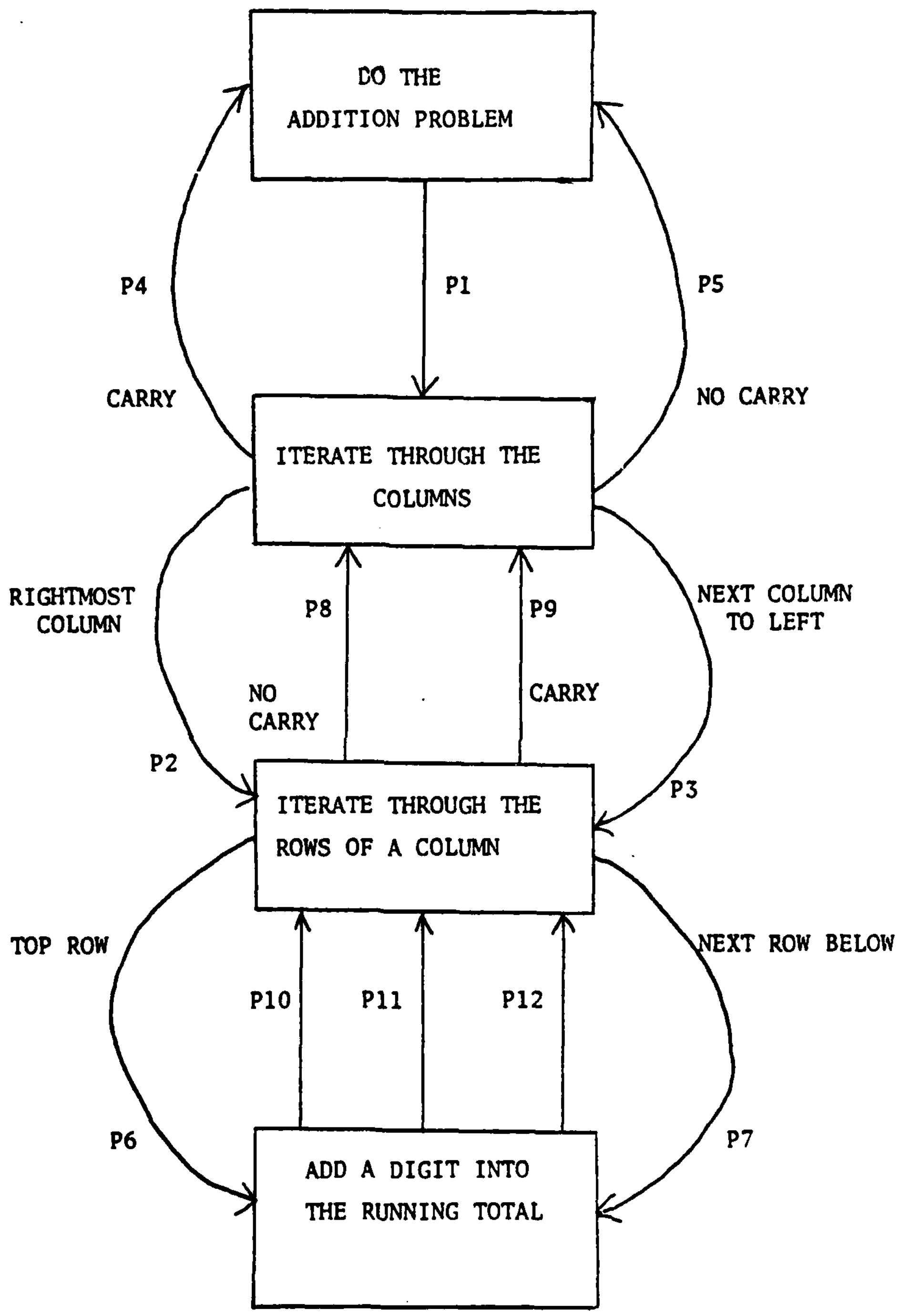


Figure 1

A representation of the flow of control in Table 1 between various goals. The boxes correspond to goal states and the arrows to productions that can change these states. Control starts with the top goal.

attention to the goal of iterating through the rows of the column. Then P7 appl
of adding 8 into the running total. P10 applies again to change the running t
create the subgoal of adding 3 into the running total; then P11 calculates the ne
point the system is back at the goal of iterating through the rows and has pro
column. Then production P9 applies which writes out the '5' in '15', sets the car
the goal of iterating through the columns. At this point the production system
the problem.

I will not trace out any further the application of this production set to the pr
to carry out the hand simulation. Note that productions P2 - P5 form a subr
columns, productions P6 - P9 an embedded subroutine for processing a colu
embedded subroutine for adding a digit to the running total. In Figure 1 all the
a subroutine emanate from the same goal box.

Significant Features of the Performance System

There are a number of features of the production system that are impor
presented. The productions themselves are the system's *procedural compon*
the clauses specified in its condition must be matched against information
information in working memory is part of the system's *declarative compone*
1980) I have discussed the network encoding of that declarative knowledg
activation defined on that network.

Goal Structure. As noted above, the productions in Table 1 are organi
subroutine is associated with a goal state that all the productions in the subro
the system can have only one goal at any moment in time, productions from
apply at any one time. This enforces a considerable seriality into the beh
seeking productions are hierarchically organized. The idea that hierarc
human cognition has been emphasized by Miller, Galanter, and Pribram (1
Van Lehn (1980) have recently introduced a similar goal-structuring fo

In the original ACT system (Anderson, 1976) there was a scheme fo
the setting of control variables. There are several important differences be
older one. First, as noted, the current scheme enforces a strong degree o
because the goals are not arbitrary nodes but rather meaningful asser
learning system to acquire new productions that make reference to goals.
be given as the various ACT learning mechanisms are discussed.

In achieving a hierarchical subroutine structure by means of a goa accepting the claim that the hierarchical control of behavior derives from the structure of problem-solving. This amounts to making the assertion that problem-solving and the goal structure it produces is a fundamental category of cognition. This is an assertion that has been advanced by others (e.g., Newell, 1980). Thus, this learning discussion contains a rather strong presupposition about the architecture of cognition. I think the presupposition is too abstract to be defended directly; rather, evidence for it will come from the fruitfulness of the systems that we can build based on the architectural assumption.

Conflict Resolution. Every production system requires some rules of conflict resolution--that is, principles for deciding which of those productions that match will be executed. ACT has a set of conflict resolution principles which can be seen as variants of the 1976 ACT or in the OPS system (Forgy & McDermott, 1977). One powerful principle is refractoriness--that the same production cannot apply to the same data in working memory twice in the same way. This prevents the same production from repeating over and over again and was implicit in the preceding hand simulation of Table 1.

The two other principles of conflict resolution in ACT are *specificity* and *strength*. Neither was illustrated in Table 1 but both are important to understanding the learning discussion. If two productions can apply and the condition of one is more specific than the other, then the more specific production takes precedence. Condition A is more specific than Condition B if the set of situations in which condition A can match is a proper subset of the set of situations where Condition B can match. The specificity rule allows exceptions to general rules to apply because these exceptions will have more specific conditions. For instance, suppose we had the following pair of productions:

PA: IF the goal is to generate the plural of man
THEN say 'MEN'

PB: IF the goal is to generate the plural of a noun
THEN say "noun + s"

The condition of production PA is more specific than the condition of production PB and so will apply over the general pluralization rule.

Each production has a strength which reflects the frequency with which that production has been successfully applied (principles for deciding if a production is successful will be given later). I will describe the rules which determine strength accumulation later in this paper; here I will describe the role of production strength in conflict resolution. Elsewhere (e.g., Anderson, 1976; Anderson, Kline, & Beasley, 1979) we have given a version of this role of strength that assumes discrete time intervals. Here I will give a continuous version. Productions are indexed by the constants in their conditions. For instance, the production PA above would be indexed by *plural* and *man*. If these concepts are active in working memory the production will be selected for consideration. In this way ACT can focus its attention on just the subset of productions that might be potentially relevant. Only if a production is selected is a test made to see if its condition is satisfied. (For future reference if a production is selected, it is said to be on the *APPLYLIST*.) A production takes a time T_1 to be selected and another time T_2 to be tested and to apply. The selection time T_1 varies with the production's strength while the application time is a constant over productions. It is further assumed that the time T_1 for the production to be selected will randomly vary from selection to selection. The expected time is a/s where s is the production strength and *a* is a constant. Although there are no compelling reasons for making any assumption about the distribution we have assumed that T_1 has an exponential distribution and attention to the goal of iterating through the rows of the column. Then P7 applie of adding 8 into the running total. P10 applies again to change the running to create the subgoal of adding 3 into the running total; then P11 calculates the new point the system is back at the goal of iterating through the rows and has proc column. Then production P9 applies which writes out the '5' in '15', sets the carr the goal of iterating through the columns. At this point the production system the problem.

I will not trace out any further the application of this production set to the pro to carry out the hand simulation. Note that productions P2 - P5 form a subrou columns, productions P6 - P9 an embedded subroutine for processing a colu embedded subroutine for adding a digit to the running total. In Figure 1 all the a subroutine emanate from the same goal box.

Significant Features of the Performance System

There are a number of features of the production system that are importa presented. The productions themselves are the system's *procedural compon* the clauses specified in its condition must be matched against information a information in working memory is part of the system's *declarative compone* 1980) I have discussed the network encoding of that declarative knowledg activation defined on that network.

Goal Structure. As noted above, the productions in Table 1 are organi subroutine is associated with a goal state that all the productions in the subro the system can have only one goal at any moment in time, productions from apply at any one time. This enforces a considerable seriality into the beh seeking productions are hierarchically organized. The idea that hierarc human cognition has been emphasized by Miller, Galanter, and Pribram (1 Van Lehn (1980) have recently introduced a similar goal-structuring fo pro

In the original ACT system (Anderson, 1976) there was a scheme for ach the setting of control variables. There are several important differences be older one. First, as noted, the current scheme enforces a strong degree o because the goals are not arbitrary nodes but rather meaningful asser learning system to acquire new productions that make reference to goals. be given as the various ACT learning mechanisms are discussed.

In achieving a hierarchical subroutine structure by means of a goa attention to the goal of iterating through the rows of the column. Then P7 applies which sets the new subgoal of adding 8 into the running total. P10 applies again to change the running total to 12; then P7 applies to create the subgoal of adding 3 into the running total; then P11 calculates the new running total as 15. At this point the system is back at the goal of iterating through the rows and has processed the bottom row of the column. Then production P9 applies which writes out the '5' in '15', sets the carry to the '1', and POP back to the goal of iterating through the columns. At this point the production system has processed one column of the problem.

I will not trace out any further the application of this production set to the problem but the reader is invited to carry out the hand simulation. Note that productions P2 - P5 form a subroutine for iterating through the columns, productions P6 - P9 an embedded subroutine for processing a column, productions P10 - P12 an embedded subroutine for adding a digit to the running total. In Figure 1 all the productions corresponding to a subroutine emanate from the same goal box.

Significant Features of the Performance System

There are a number of features of the production system that are important for the learning theory to be presented. The productions themselves are the system's *procedural component.* For a production to apply, the clauses specified in its condition must be matched against information *active* in working memory. This information in working memory is part of the system's *declarative component.* Elsewhere (Anderson, 1976, 1980) I have discussed the network encoding of that declarative knowledge and the process of spreading activation defined on that network.

Goal Structure. As noted above, the productions in Table 1 are organized into subroutines where each subroutine is associated with a goal state that all the productions in the subroutine are trying to achieve. Since the system can have only one goal at any moment in time, productions from only one of these subroutines can apply at any one time. This enforces a considerable seriality into the behavior of the system. These goal-seeking productions are hierarchically organized. The idea that hierarchical structure is fundamental to human cognition has been emphasized by Miller, Galanter, and Pribram (1960) and many others. Brown and Van Lehn (1980) have recently introduced a similar goal-structuring for production systems.

In the original ACT system (Anderson, 1976) there was a scheme for achieving the effect of subroutines by the setting of control variables. There are several important differences between the current scheme and that older one. First, as noted, the current scheme enforces a strong degree of seriality into the system. Second, because the goals are not arbitrary nodes but rather meaningful assertions, it is much easier for ACT's learning system to acquire new productions that make reference to goals. Evidence for this last assertion will be given as the various ACT learning mechanisms are discussed.

In achieving a hierarchical subroutine structure by means of a goal-subgoal structure, I am of course accepting the claim that the hierarchical control of behavior derives from the structure of problem-solving. This amounts to making the assertion that problem-solving and the goal structure it produces is a fundamental category of cognition. This is an assertion that has been advanced by others (e.g., Newell, 1980). Thus, this learning discussion contains a rather strong presupposition about the architecture of cognition. I think the presupposition is too abstract to be defended directly; rather, evidence for it will come from the fruitfulness of the systems that we can build based on the architectural assumption.

Conflict Resolution. Every production system requires some rules of conflict resolution--that is, principles for deciding which of those productions that match will be executed. ACT has a set of conflict resolution principles which can be seen as variants of the 1976 ACT or in the OPS system (Forgy & McDermott, 1977). One powerful principle is refractoriness--that the same production cannot apply to the same data in working memory twice in the same way. This prevents the same production from repeating over and over again and was implicit in the preceding hand simulation of Table 1.

The two other principles of conflict resolution in ACT are *specificity* and *strength*. Neither was illustrated in Table 1 but both are important to understanding the learning discussion. If two productions can apply and the condition of one is more specific than the other, then the more specific production takes precedence. Condition A is more specific than Condition B if the set of situations in which condition A can match is a proper subset of the set of situations where Condition B can match. The specificity rule allows exceptions to general rules to apply because these exceptions will have more specific conditions. For instance, suppose we had the following pair of productions:

PA: IF the goal is to generate the plural of man
THEN say 'MEN'

PB: IF the goal is to generate the plural of a noun
THEN say "noun + s"

The condition of production PA is more specific than the condition of production PB and so will apply over the general pluralization rule.

Each production has a strength which reflects the frequency with which that production has been successfully applied (principles for deciding if a production is successful will be given later). I will describe the rules which determine strength accumulation later in this paper; here I will describe the role of production strength in conflict resolution. Elsewhere (e.g., Anderson, 1976; Anderson, Kline, & Beasley, 1979) we have given a version of this role of strength that assumes discrete time intervals. Here I will give a continuous version. Productions are indexed by the constants in their conditions. For instance, the production PA above would be indexed by *plural* and *man*. If these concepts are active in working memory the production will be selected for consideration. In this way ACT can focus its attention on just the subset of productions that might be potentially relevant. Only if a production is selected is a test made to see if its condition is satisfied. (For future reference if a production is selected, it is said to be on the *APPLYLIST*.) A production takes a time T_1 to be selected and another time T_2 to be tested and to apply. The selection time T_1 varies with the production's strength while the application time is a constant over productions. It is further assumed that the time T_1 for the production to be selected will randomly vary from selection to selection. The expected time is a/s where s is the production strength and *a* is a constant. Although there are no compelling reasons for making any assumption about the distribution we have assumed that T_1 has an exponential distribution and this is its form in all our simulations.

A production will actually apply if it is selected and it has completed application before a more specific production is selected. This provides the relationship between strength and specificity in our theory. A more specific production will take precedence over a more general production only if its selection time is less than the selection and application time of the more general production. Since strength reflects frequency of practice, only exceptions that have some criterion frequency will be able to reliably take precedence over general rules. This corresponds, for instance, to the fact that words with irregular inflections tend to be of relatively high frequency. It is possible for an exception to be of borderline strength so that it sometimes is selected in time to beat out the general rule but sometimes not. This corresponds, for instance, to the stage in language development when an irregular inflection is being used with only partial reliability (Brown, 1973).

Variables. Productions contain variable slots which can take on different values in different situations. The use of these variables is often implicit, as in Table 1, but sometimes it is important to acknowledge the variables that are being assumed. As an illustration, let us consider a variabilized form of a production from Table 1. If production P9 from that table were to be written in a way to expose its variable structure, it would have the form below where the terms prefixed by 'LV' are local variables:

```
    IF the goal is to iterate through the rows of LVcolumn
       and LVrow is the last row of LVcolumn
       and LVrow has been processed
       and the running total is of the form "LVstring + LVdigit"
  THEN write LVdigit
       and set carry to LVstring
       and mark LVcolumn as processed
       and POP the goal
```

Local variables can be reassigned to new values each time the production applies. Thus, for instance, the terms LVcolumn, LVrow, LVstring, and LVdigit will match to whatever elements lead to a complete match of the condition to working memory. Suppose, for instance, that the following elements were in working memory:

The goal is to iterate through the rows of column-2
Row-x is the last row of column-2
Row-x has been processed
Running total is of the form 2 + 4

The production would match this working-memory information with the following variable binding:

LVcolumn = column-2
LVrow = row-x
LVstring = 2
LVdigit = 4

Local variables assume values within a production for the purposes of matching the condition and executing the action. After application of the production variables lose their values.

Learning in ACT

This paper is concerned with the processes underlying the acquisition of cognitive skill. As is clear from examples like Table 1 there is a closer connection in ACT between productions and skill performance than between declarative knowledge and skill performance. This is because the control over cognition and behavior lies directly in the productions. Facts are used by the productions. So, in a real sense facts are instruments of the productions which are the agents. For instance, we saw that production P10 used the addition fact that "4 + 8 = 12". Although productions are closer to performance than facts, I will be claiming that when a person initially learns about a skill he learns only facts about the skill and does not directly acquire productions. These facts are used interpretively by general-purpose productions. The first major section of this paper, on the *declarative stage*, will both discuss the evidence for the claim that initial learning of a skill involves just acquisition of facts and explain how general-purpose productions can interpret these facts to generate performance of the skill.

The next major section of the paper will discuss the evidence for and nature of the knowledge compilation process which results in the translation from a declarative base for a skill to a procedural base for the skill. (For instance, the production set in Table 1 is a procedural base for the addition skill.) After this section, the remainder of the paper will discuss the continued improvement of a skill after it has achieved a procedural embodiment. In all of this I will be drawing heavily on the work we have done studying the acquisition of proof skills in geometry (Anderson, Greeno, Kline, & Neves, 1981; Neves & Anderson, 1981).

The Declarative Stage: Interpretive Procedures

One of the things that becomes apparent in studying the initial stages of skill acquisition in areas of mathematics like geometry or algebra (e.g., Neves, 1981) is that the instruction seldom if ever directly specifies a procedure to be applied. Still, the student is able to emerge from this type of instruction with an ability to generate behavior that reflects knowledge contained in the instruction. Figures 2, 3, and 4 from our work on geometry illustrate this point. Figure 2 is taken from the text of Jurgensen, Donnelly, Maier, & Rising (1975) and represents the total of that text's instruction on two-column proofs. Immediately after studying this, two of our students attempted to give reasons for two-column proof problems. The first such proof problem is the one illustrated in Figure 3. Both of the students were able to deal with this problem with some success.

1-7 *Proofs in Two-Column Form*

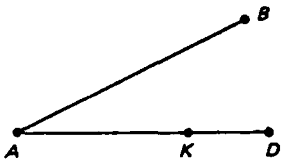
You prove a statement in geometry by using deductive reasoning to show that the statement follows from the hypothesis and other accepted material. Often the assertions made in a proof are listed in one column, and reasons which support the assertions are listed in an adjacent column.

EXAMPLE. A proof in two-column form.

Given: $\overline{AKD}$; $AD = AB$

Prove: $AK + KD = AB$

Proof:



STATEMENTS	REASONS
1. $\overline{AKD}$	1. Given
2. $AK + KD = AD$	2. Definition of between
3. $AD = AB$	3. Given
4. $AK + KD = AB$	4. Transitive property of equality

Some people prefer to support Statement 4, above, with the reason *The Substitution Principle*. Both reasons are correct.

The reasons used in the example are of three types: *Given* (Steps 1 and 3), *Definition* (Step 2), and *Postulate* (Step 4). Just one other kind of reason, *Theorem*, can be used in a mathematical proof. Postulates and theorems from both algebra and geometry can be used.

Reasons Used in Proofs

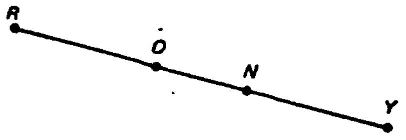
Given (Facts provided for a particular problem)
Definitions
Postulates
Theorems that have already been proved.

Figure 2: The text instruction in two column proof.

Given: $\overline{RONY}$; $\overline{RO} \cong \overline{NY}$

Prove: $RN = OY$

Proof:



STATEMENTS	REASONS
1. $\overline{RO} \cong \overline{NY}$	1. ?
2. $RO = NY$	2. ?
3. $ON = ON$	3. ?
4. $RO + ON = ON + NY$	4. ?
5. $\overline{RONY}$	5. ?
6. $RO + ON = RN$	6. ?
7. $ON + NY = OY$	7. ?
8. $RN = OY$	8. ?

Figure 3: A reason-giving task that is the first problem that the student encounters requiring use of the knowledge about two column proofs.

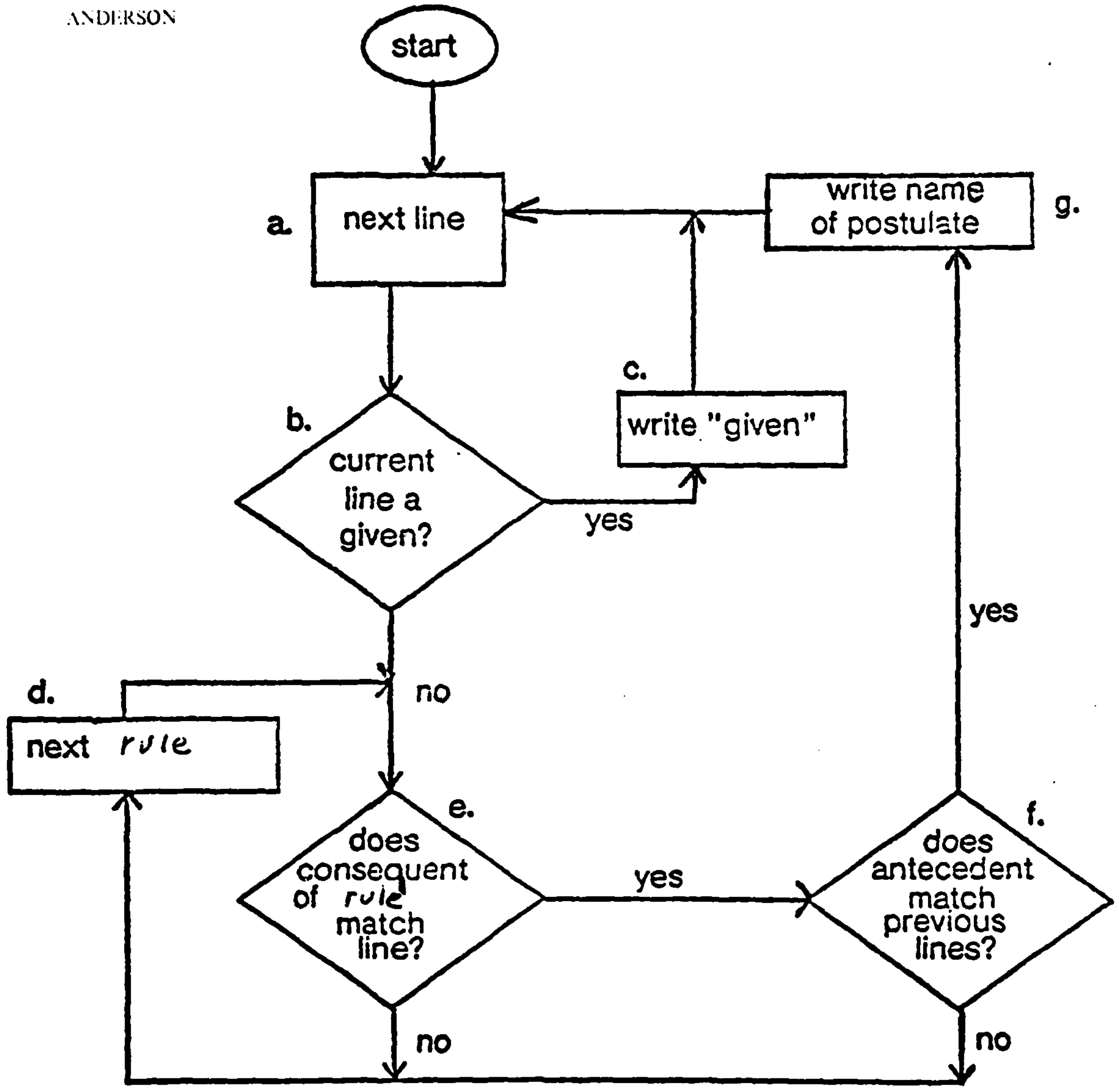


Figure 4: A flowchart showing the general flow of control in a reason-giving task.

Behavior on these reason-giving problems is rather constant across subjects at least at a global level. Figure 4 is a representation at the global level of these constancies. Clearly, there is nowhere in Figure 2 a specification of the flow of control that is in Figure 4. However, before reading the instruction of Figure 2 subjects were not capable of the flow of control in Figure 4 and after reading the instruction they were. So somehow the instruction in Figure 2 makes the procedure in Figure 4 possible.

Two of the ways that students bridge the gap between inadequate instruction and behavior are:

1. Use of general problem solving skills and prior knowledge to fill in the missing pieces and resolve the ambiguities.

2. Analogy. One variant on the analogy method is that students use worked-out examples of solutions to problems as models for solving a current problem. Another variant on this method is that students use a prior procedure that does something analogous to the desired behavior and try to modify the output of this procedure.

It is characteristic of both the problem-solving possibility and the analogy possibility that the domain knowledge is being used by domain-independent general procedures. For this reason I say that the knowledge about the skill is being used *interpretively.* The term reflects the fact that the knowledge is data for other procedures in just the way a computer program is data for an interpreter. In the two subsections to follow I will discuss examples of how behavior can be generated by means of application of general problem solving methods and by means of analogy. These examples serve three functions. First, they make concrete how task-appropriate behavior can be generated without task-specific procedures. Second, in relating these examples to data from our protocols, I will be able to give additional empirical support for the claim that a skill starts out initially in a declarative state. Third, by explaining the initial character of skill organization, I will be laying the foundation for explanation of later learning mechanisms.

It is a strong claim that all skill learning starts with the declarative encoding of facts about the skill domain. The learning in the declarative stage, then, is the same kind of learning that occurs when a student reads a story or memorizes a paired-associate list. From the point of view of understanding skill acquisition, this is rather trivial learning. Part of its virtue is that it is trivial--that it does not require elaborate self-understanding on the part of the student.

Application of General Problem Solving Methods

Even though the student coming upon the instruction in Figure 2 has no procedures specific to doing two column proof problems, he has procedures for solving problems in general, for doing mathematics-like exercises, and perhaps even for certain types of deductive reasoning. These general problem-solving procedures can use the instruction such as that in Figure 2 as data for generating task-appropriate behavior when faced with a problem like that in Figure 3. Below is a review of a simulation of how this can happen.

An Example. Table 2 provides a listing of the productions used in this simulation and Figure 5 illustrates their flow of control. It is assumed that the student encodes the exercise in Figure 3 as a list of subproblems where each subproblem is to write a reason for a line of the proof. If so, production P1 applies first and it focuses attention on the first subproblem--that is, it sets as the subgoal to write a reason for $\overline{RO} \cong \overline{NY}$. Next production P4 applies. P4's condition, "the goal is to write the name of a relation for an argument," matches the current subgoal "to write the name of the reason for $\overline{RO} \cong \overline{NY}$". P4 creates the subgoal of finding a reason for the line. P4 is quite general and reflects the existence of a prior procedure for writing statements that satisfy a constraint.

The student presumably has encoded the boxed information in Figure 2 as indicating a list of methods for providing a reason for a line. If so production P7 applies next and sets as a subgoal to try givens, the first rule on the reason list, as a justification for the current line. Note this is one point where a fragment of the

Table 2

Interpretive Productions Evoked in Performing the Reason-Giving Task

```
P1:        IF the goal is to do a list of problems
         THEN set as a subgoal to do the first problem in the list

P2:        IF the goal is to do a list of problems
              and a problem has just been finished
         THEN set as a subgoal to do the next problem

P3:        IF the goal is to do a list of problems
              and there are no unfinished problems on the list
         THEN POP the goal with success

P4:        IF the goal is to write the name of a relation for an argument
         THEN set as a subgoal to find what the relation is for the argument

P5:        IF the goal is to write the name of a relation for an argument
              and a name has been found
         THEN write the name
              and POP the goal with success

P6:        IF the goal is to write the name of a relation for an argument
              and no name has been found
         THEN POP the goal with failure

P7:        IF the goal is to find a relation
              and there is a list of methods for achieving the relation
         THEN set as a subgoal to try the first method

P8:        IF the goal is to find a relation
              and there is a list of methods for achieving the relation
              and a method has just been unsuccessfully tried
         THEN set as a subgoal to try the next method

P9:        IF the goal is to find a relation
              and there is a list of methods for achieving the relation
              and a method has been successfully tried
         THEN POP the goal with success

P10:       IF the goal is to find a relation
              and there is a list of methods for achieving the relation
              and they have all proven unsuccessful
         THEN POP the goal with failure
```

```
P11:      IF the goal is to try a method
             and that method involves establishing a relationship
       THEN set as a subgoal to establish the relationship

P12:      IF the goal is to try a method
             and the subgoal was a success
       THEN POP the goal with success

P13:      IF the goal is to try a method
             and the subgoal was a failure
       THEN POP the goal with failure

P14:      IF the goal is to establish that a statement is among a list
             and the list contains the statement
       THEN POP the goal with success

P15:      IF the goal is to establish that a statement is among a list
             and the list does not contain the statement
       THEN POP the goal with failure

P16:      IF the goal is to establish that a line is implied by a rule in a set
             and the set contains a rule of the form consequent if antecedents
             and the consequent matches the line
       THEN set as a subgoal to determine if the antecedents correspond to established statements
             and tag the rule as tried

P17:      IF the goal is to establish that a line is implied by a rule in a set
             and the set contains a rule of the form consequent if antecedents
             and the consequent matches the line
             and the antecedents have been established
       THEN POP the goal with success

P18:      IF the goal is to establish that a line is implied by a rule in a set
             and there is no untried rule in the set which matches the line
       THEN POP the goal with failure

P19:      IF the goal is to determine if antecedents correspond to established statements
             and there is an unestablished antecedent clause
             and the clause matches an established statement
       THEN tag the clause as established

P20:      IF the goal is to determine if antecedents correspond to established statements
             and there are no unestablished antecedent clauses
       THEN POP the goal with success

P21:      IF the goal is to determine if antecedents correspond to established statements
             and there is an unestablished antecedent clause
             and it matches no established statement
       THEN POP the goal with failure
```

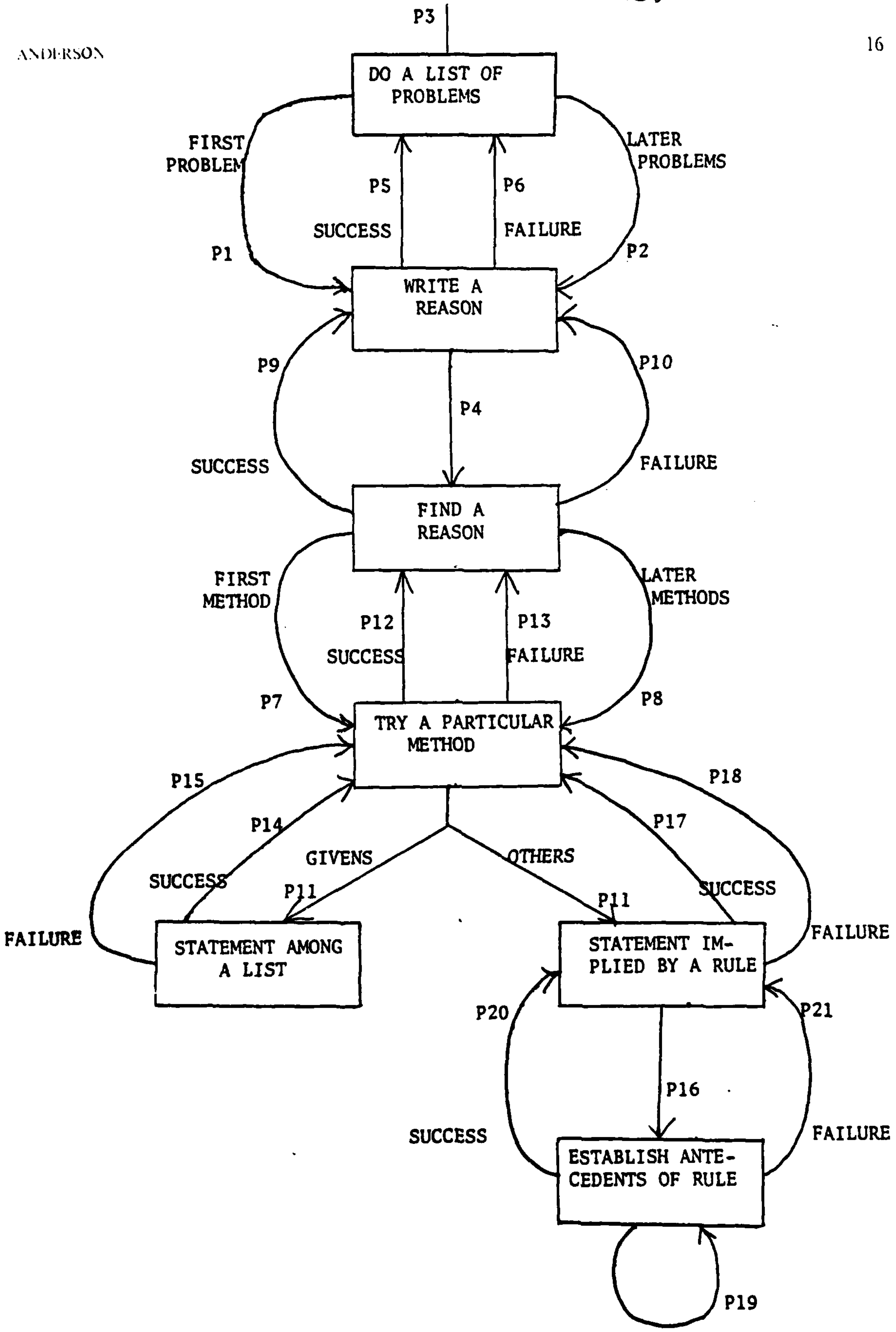


Figure 5: A representation of the flow of control in Table 2 between the various goals. Control starts with the top goal.

instruction is used by a general problem-solving procedure (in this case, for searching a list of methods) to determine the course of behavior.

The students we studied had extracted from the instruction in Figure 2 that the *givens* reasons is used when the line to be justified is among the givens of the problem. Note that this fact is not explicitly stated in the instruction but is strongly implied. Thus, it is assumed that the student has encoded the fact that "the givens method involves establishing that the statement is among the givens." Production P11 will match this fact in its condition and so will set as a subgoal to establish that RO = NY is among the givens of the problem. Production P14 models the successful recognition that RO = NY is among the givens and returns a success from the subgoal. That is to say, its action "POP the goal with success" tags the goal "to find RO = NY among the givens" with success and sets as the current goal the higher goal of trying the givens method. Then P12 and P9 POP success back up to the next-to-top-level-goal of writing a reason for the line. Then production P5 applies to write "given" as a reason and POPs back to the top-level goal.

At this point production P2 applies to set the subgoal of writing a reason for the second line, RO = NY. Then productions P4, P7, and P11 apply in that order setting the subgoal of seeing whether RO = NY was among the givens of the problem. Production P15 recognizes this as a failed goal and then production P13 returns control back to the level of choosing methods to establish a reason. Production P8 selects the definition reason next to try.

Clearly, the instruction in Figure 2 contains no explanation of how a definition should be applied. However, the assumption of the text is that the student knows that a definition should imply the statement. There were some earlier exercises on conditional and biconditional statements that makes this assumption at least conceivable. Our two subjects both knew that some inference-like activity was required but they had a faulty understanding of the nature of the application of inference to this task. In any case, assuming that the student knows as a *fact* (in contrast to a procedure) that use of definitions involves inferential reasoning, production P11 will match in its condition the fact that "definitions involve establishing that the statement is implied by a definition" and P11 will set the subgoal of proving that RO = NY was implied by a definition.

At this point I have to momentarily leave our students behind and describe the ideal student. The textbook assumes that the student already has a functioning procedure for finding a rule that implies a statement by means of a set of established rules. Productions P16 - P21 constitute such a procedure. Neither of our students had this procedure in its entirety. These productions work as a general inference testing procedure and apply equally well to postulates and theorems as well as definitions. Production P16 selects a conditional rule that matches the current line (the exact details of the match are not unpacked in Table 2). It is assumed that a biconditional definition is encoded as two implications each of the form consequent if antecedent. The definition relevant to the current line 2 is that two line segments are congruent if and only if they are of equal measure which is encoded as:

$$\underline{XZ} = \underline{UV} \text{ if } \overline{XZ} \cong \overline{UV}$$
$$\text{and } \overline{XZ} \cong \overline{UV} \text{ if } XZ = UV$$

the first implication is the one that is selected and the subgoal is set to establish the antecedent $\overline{XZ} \cong \overline{UV}$ (or $\overline{RO} \cong \overline{NY}$, in the current instantiation). The production set P19 - P21 describe a procedure for matching zero or more clauses in the antecedent of a rule. In this case P19 finds a match to the one condition, $\overline{XZ} \cong \overline{UV}$, with $\overline{RO} \cong \overline{NY}$ in the first line. Then P20 pops with success followed by successful popping of P17, then P12, and then P9 which returns the system to the goal of writing out a reason for the line.

Significant Features of the Example. I will not further trace the application of the production set to the example. I would like to identify, however, the essential aspects of how this production set allows the student to bridge the gap between instruction and the problem demands. Figure 5 illustrates the flow of control with each box being a level in the goal structure and serving as subroutine. Although it is not transparent, the subgoal organization in Figure 5 results in the same flow of control as the flowchart organization of Figure 4. However, as the production rendition of Figure 5 establishes, the flow of control in Figure 5 is not something fixed ahead of time but rather emerges in response to the instruction and the problem statement.

The top level goal in Figure 5 of iterating through a list of problems is provided by the problem statement and, given the problem statement, it is unpacked into a set of subgoals to write statements indicating the reasons for each line. This top level procedure reflects a general strategy the student has for decomposing problems into linearly ordered subproblems. Then another prior routine sets as subgoals to find the reasons. At this point the instruction about the list of acceptable relationships is called into play (through yet another prior problem-solving procedure) and is used to set a series of subgoals to try out the various possible relationships. So the unpacking of subgoals in Figure 5 from "do a list of problems" to "find a reason" is in response to the problem statement; the further unpacking into the methods of givens, postulates, definitions, and theorems is in response to the instruction. The instruction is the source of information identifying that the method of givens involves searching the given list and the other methods involve application of inferential reasoning. The ability to search a list for a match is assumed by the text, reasonably enough, as a prior procedure on the part of the student. The ability to apply inferential reasoning is also assumed as a prior procedure, but in this case the assumption is mistaken.

In summary, then, we see in Figure 5 a set of separate problem-solving procedures which are joined together in a novel combination in response to the problem statement and instruction. In this sense, the student's general problem-solving procedures are interpreting the problem statement and instruction. Note that the problem statement and the instruction are being brought into play by being matched as data in the conditions of the productions of Table 2.

Student Understanding of Implication. The two students that we studied both had serious misunderstandings about how one determines if a statement is implied by a rule and we spent some time correcting each student's misconceptions. One student thought that it was sufficient to determine that the consequent of the rule matched the to-be-justified statement and did not bother to test the antecedent. For him, the subroutine call (subgoal setting) of production P16 did not exist.

Our second student had more exotic misunderstandings. This is best illustrated in his efforts to justify the line 4, RO + ON = ON + NY, in Figure 3. The student thought the transitive property of equality was the right justification for line 4. The transitive property of equality is stated as "a = b, b = c, implies a = c."

The student physically drew out the following correspondence between the antecedents of this postulate and the to-be-justified statement:

$$\begin{tikzcd}[column sep=small] a \arrow[d, dash] & = & b, \arrow[d, dash] & & b \arrow[d, dash] & = & c \arrow[d, dash] \\ RO & + & ON & = & ON & + & NY \end{tikzcd}$$

That is, he found that he could put the variables of the antecedent in order with the terms of the statement. He noted that he needed to also match to a = c in the consequent of the transitive postulate but noted that a previous line had RO = NY, which given the earlier variable matches, satisfied his need.

This student had at least two misunderstandings. First, he seemed unable to appreciate the tight constraints on pattern matching (e.g., one cannot match "=" against "+"). Second, he failed to appreciate that the consequent of the postulate should be matched to the statement and the antecedent to earlier statements. Rather he had it the other way around. However, given the instruction he has had to date this is not surprising since none of this was specified.

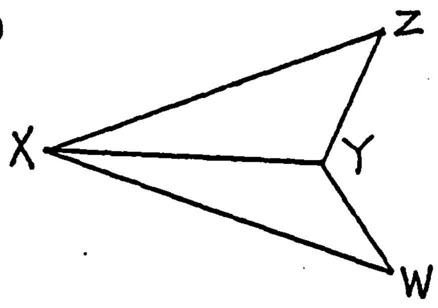
Both students required remedial instruction. Thus, these errors created the opportunity for new learning. Although I have not analyzed this in detail, I believe that remedial instruction amounted to providing additional declarative information. This information could be used by other general procedures to provide interpretive behavior in place of the compiled procedures that Table 2 is assuming in productions P16 - P21. This is a simple form of debugging: When the instruction assumes pre-compiled procedures which do not exist, remedial instruction can correct the situation by providing the data for interpretive procedures.

Use of Analogy

In the previous discussion we saw how general problem solving procedures can be used in the absence of specific procedures. Such procedures generate behavior in response to the specifications of the problem and the constraints of the instruction. Given the incompleteness of the instruction, this way of solving problems is often a very difficult and sometimes impossible route to follow. An alternative is to try to generate the requisite behavior out of analogy to a model of the correct performance. We will consider two sources for such a model. One is the worked-out examples found in instruction and the other is the product of similar procedures that one possesses. In both cases the analogy is being carried out by general procedures using the example as data. In both uses, success of the analogy process depends on how well the analogy process is informed of the constraints of the domain. That is, success is seldom possible by means of a blind symbol-for-symbol substitution from the example to the new problem. The analogy process must take advantage of the instruction in order to perform the mapping intelligently.

Analogy to Examples. Figure 6 illustrates a case of successful use of analogy in our protocol, but is somewhat exceptional in that it is a case where an almost pure symbol-for-symbol mapping seems to work. It does serve to illustrate, however, the basic features of analogy to example. The first problem is presented in the text as a proof for which reasons have to be given. I have given the problem in Figure 6a with the reasons

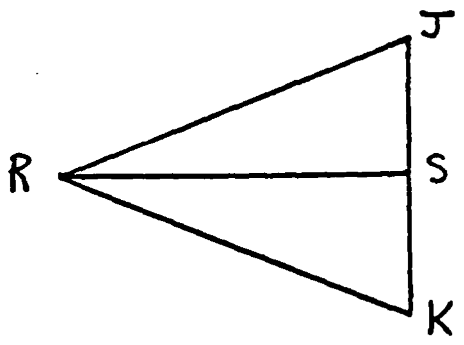
(a)



Given: $\overline{XZ} \cong \overline{XW}$, $\overline{YZ} \cong \overline{YW}$
Prove: $\triangle XYZ \cong \triangle XYW$

Statements	Reasons
$\overline{XZ} \cong \overline{XW}$	Given
$\overline{XY} \cong \overline{XY}$	Reflexive Property of Congruence
$\overline{YZ} \cong \overline{YW}$	Given
$\triangle XYZ \cong \triangle XYW$	SSS

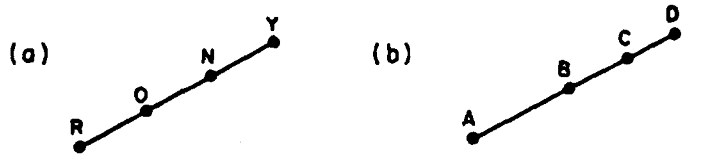
(b)



Given: $\overline{RJ} \cong \overline{RK}$, $\overline{SJ} \cong \overline{SK}$
Prove: $\triangle RSJ \cong \triangle RSK$

Figure 6: Problem (b) is easily solved by analogy to Problem (a).

provided. The second problem was presented as the first proof-generation problem of the section. Our subject immediately noticed the analogy to the prior example problem and went about copying over the proof with the appropriate modifications made. This example represents the essential two-stage process involved in using analogy to prior examples. There is first a process of detecting a similarity between two problem situations. Second, there is the process of deciding if and how the correspondence is to be used. The similarity is detected by the partial match between the two problem descriptions. The model of this partial-matching process (described in Kline, 1981) basically counts up the commonalities between the two problem descriptions and can be quite influenced by superficial similarities such as orientation of the diagrams--as human students are. Once the similarity is detected an attempt is made to map the analogy from one problem to the next. The mapping is very easy in this case: X -> R, Z -> J, Y -> S, W -> K.



Given: RO = NY, $\overline{RONY}$	Given: AB > CD, $\overline{ABCD}$
Prove: RN = OY	Prove: AC > BD
RO = NY	AB > CD
ON = ON	BC > BC
RO + ON = ON + NY	
$\overline{RONY}$	! ! !
RO + ON = RN	
ON + NY = OY	
RN = OY	

Figure 7: One student ran into difficulty trying to use the proof in (a) as an analogy for generating a proof for (b). See text for discussion.

In other cases, the correspondence is not so easy. Figure 7 illustrates another attempt on our subject's part to use analogy. Part a of the Figure illustrates the same reason-giving problem that we have already seen as part of Figure 3. Part b illustrates our student's start at generating a proof to a later problem in the section. He noted the obvious similarity between the two problem statements and proceeded to draw the analogy. Apparently, he inferred that he could simply substitute elements for elements and tried the following mapping: R -> O, O -> B, N -> C, Y -> D, and equal -> inequality. This allows for a complete mapping of one problem statement onto the other. With these mappings he then tried to copy over the proof. He got the first line correct: Analogous to RO = NY he wrote AB > CD. Then he had to write something analogous to ON = ON. He wrote BC > BC! Almost immediately his semantic sensitivities perceived the absurdity of this statement and he simply gave up the attempt to use the analogy and turned to trying to solve the problem anew.

In such attempts at analogy, a declarative knowledge structure (the representation of the prior example) is being used interpretively. It is first used by some similarity-detecting procedure and then by the procedure that does the analogy-mapping. The analogy mapper tries to transform the steps of one problem into the steps of another. It is possible to have more sophisticated procedures for analogy mapping than those displayed by our subject for the problem in Figure 7 and occasionally our subjects displayed such sophistication in use of analogy.

Analogy to Output of a Prior Procedure. Yet another way to get successful behavior in initial situations is to select some established procedure for a domain similar to the current one and try to extend the procedure to the current domain. This use of analogy can be modelled as applying the established procedure directly to the new domain and then taking the output of this procedure as a declarative structure to be modified according to the current domain constraints. An example of this occurred in the protocols of another subject on the problem illustrated in Part b of Figure 7. Before discussing his problem attempt, it is worth first noting that this problem is very similar to problems that a student might face in contexts other than geometry. Although we did not get his protocol on such a problem, imagine how one would deal with a problem such as:

> On Labor Day both Willie Stargell and Dave Parker were hitting .300. However, Parker had more at bats. During the stretch drive after Labor Day, Stargell had 8 home runs, 8 doubles, and 12 singles. During the stretch drive, Parker had 5 home runs, 2 triples, 10 doubles, and 11 singles. Who had the most hits for the season?

The following is the protocol of a Ph.D.--presumably, we should not expect better from an eighth grader:

> Well, Stargell had 8 + 8 + 12 = 28 hits in the stretch drive and Parker had 5 + 2 + 10 + 11 = 28 hits too. Therefore, they had as many hits in the stretch drive. They both hit .300 before Labor Day but Parker had more at-bats. Therefore, Parker had more hits before Labor Day. Therefore, he had more hits for the whole season.

The basic plan for this argument can be seen to derive from a production of the sort:

```
IF the goal is to argue that $X_1 > X_2$
   and $X_1 = a_1 + b_1$
   and $X_2 = a_2 + b_2$
   and $a_1 > a_2$
   and $b_1 = b_2$
THEN set as subgoals to argue that $a_1 > a_2$
                 and to argue that $b_1 = b_2$
```

The major line of the argument in the above protocol was directed to achieving these subgoals. What if we presented a degenerate problem on the order of Figure 7b?

> Dave Parker had more hits than Stargell before Labor Day and they both had the same number of hits after Labor Day. Who had the most hits for the whole season?

Presumably, the student would have given a simple argument for this problem of the form:

> Parker had more hits before Labor Day.
> They had the same number of hits in the stretch drive.
> Therefore, Parker had more hits for the whole season.

The two subarguments are simply stated and then the conclusion stated. While this structure for an argument is quite acceptable when applied to this informal domain, it results in problems when the student tries to map it onto a geometry proof.

We believe that our subject tried to apply this established style of argumentation, as embodied in the above production, to the problem in Figure 7b. In his initial analysis of the problem he did note that segments AB and BC formed AC and that segments BC and CD formed BD. He also observed that he was given the requisite inequality and equality. So, he had available all the information required for the above production to match. If he were to translate the goals and subgoals of the argument structure into lines, he would come up with something like:

$AB > CD$

$$\frac{BC = BC}{\therefore AC > CD}$$

We speculate that he tried to map the output of this argument to the current problem by making these lines of the argument statements of the problem. He knew that an additional constraint in the geometry domain was that he had to give reasons for each of his lines. For the first two lines he quickly saw the reasons--"given" and "reflexive rule of equality"--presumably, matching against his meager knowledge of geometry and past postulates. So, what we saw from him was a quick writing down of the first two lines of the proof along with the correct justifications. At this point if he were trying to map his argument structure into a proof structure with acceptable reasons, he should come to an impasse in that there was no geometry justification that he could give for the next line of his argument, namely $AC > CD$. It was at this point he turned to the earlier problem worked out in the text--namely, problem (a) in Figure 7. He determined the analogous statement and the analogous reason for this problem. Therefore, he wrote for his third line:

	Statement	Reason
3)	$AB + BC > BC + CD$	Addition Property of Inequality

He then wrote for his fourth line:

4)	$AC > BD$	Subtraction Property of Inequality

We speculate that in this fourth line he was writing out the final line of his argument structure because he thought he saw a reason of geometry that would justify it. When we asked him why he wrote subtraction property of inequality, he pointed out that he was "subtracting out" the B from the left hand side in the inequality in (3) to get the left hand side of the inequality in (4) and similarly obtained the right-hand side of (4) by "subtracting out" the C from the right-hand side of (3).

We have been intrigued by this protocol because on one hand, the subject gave clear evidence in his preanalysis of the problem and in his choice of steps that he did have some understanding of the problem. On the other hand, his error reflects such a gross misunderstanding of the domain. Our speculation is that his understanding derives from applying this past argument structure to the problem and that his misunderstanding derives from trying to map this argument structure into geometry. The problem is that the constraints on an acceptable proof are much stronger than those on an acceptable verbal argument. Specifically, it is necessary to make explicit and justify the assumptions that $AB + BC = AC$ and $BC + CD = BD$. Note in our Ph.D.'s protocol that there was no attempt to explicitly state or justify the analogous assumptions that number of hits for the whole year are the sum of the number of hits before Labor Day and the number of hits in the stretch drive. This is tested for in the condition of the argument production, it is just not made an explicit part of the argument by the action of this production.

Use of Analogy: Summary. By way of summary Figure 8 illustrates the two paths we have considered for application of analogy. Both start from the statement of the problem. The first detects some similarity between the problem-statement and the statement of another problem. The solution to this previous problem is retrieved (often by looking it up) and an attempt is made to map this solution onto a solution to the current problem. This mapping is very much like solving the classic analogy syllogism--i.e., the student solves "The prior problem statement is to the current problem statement as the prior solution is to ?" The other possibility is that the problem statement will evoke some other procedure. The application of this procedure's production will result in a solution. Again the student must map this solution onto a solution for the current problem but now he cannot treat this as an analogy syllogism and there is less to guide his attempt to map the solution.

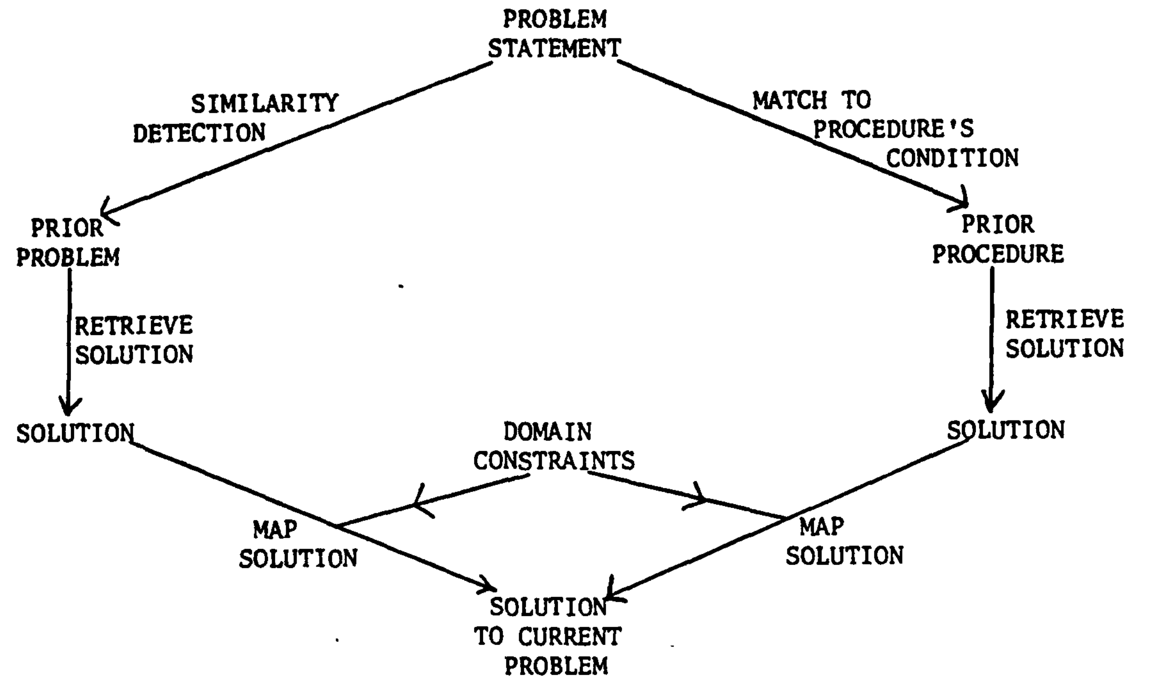


Figure 8: Illustration of the steps of processing in the two types of analogy use.

In either case the mapping process will get into trouble if the student does not observe the constraints of the geometry proof domain. We saw evidence for such problems in our subject protocols. The teacher in instructing the student can use these failures as opportunities to reinforce the domain constraints through remedial instruction. On can imagine that the subjects would develop serious misunderstandings without such teacher feedback. Brown and Van Lehn (1980) speculate that many subtraction bugs that children possess derive from self attempts to repair problems in their procedures.

We have placed our discussion of analogy under the heading of interpretive use of declarative knowledge. The declarative knowledge that is being used in analogy is the solution to be mapped and domain constraints on the mapping. The procedures that do the mapping are the interpretive procedures.

The Need for an Initial Declarative Encoding

This section has been concerned with showing how students can generate behavior in a new domain when they do not have specific procedures for acting in that domain. Their knowledge of the domain is declarative and is interpreted by general procedures. One can argue that it is adaptive for a learning system to start out this way. New productions have to be integrated with the general flow of control in the system. Clearly we are not in possession of an adequate understanding of our flow of control to form such productions directly. One of the reasons why instruction is so inadequate is that the teacher likewise has a poor conception of flow of control in the student. Attempts to directly encode new procedures, as in the Instructible Production System (Rychener, 1981; Rychener & Newell, 1978), have run into trouble because of this problem of integrating new elements into a complex existing flow of control.

As an example of the problem with creating new procedures out of whole cloth, consider the use of the definition of congruence by the production set in Table 2 to provide a reason for the second line in Figure 3. One could build a production that would directly recognize the application of the definition to this situation rather than going through the interpretive rigamarole of Figure 5 (Table 2). This production would have the form:

```
    IF the goal is to give a reason for $XY = UV$
          and a previous line has $\overline{XY} \cong \overline{UV}$
THEN POP with success
          and the reason is definition of segment congruence
```

However, it is very implausible that the subject could know that this knowledge was needed in this procedural form before he stumbled on its use to solve line 2 in Figure 3. Thus, ACT should not be expected to encode its knowledge into procedures until it has seen examples of how the knowledge is to be used.

While new productions have to be created sometimes, forming new productions is potentially a dangerous thing. Because productions have direct control over behavior there is the ever present danger that a new production may wreak great havoc in a system. Anyone who incrementally augments computer programs will be aware of this problem. A single erroneous statement can destroy the behavior of a previously fine program. In computer programming the cost is slight--one simply has to edit out the bugs the new procedure brought in. For an evolving creature the cost of such a failure might well be death. In the next section we will describe a highly conservative and adaptive way of entering new procedures.

As the examples reviewed in this section illustrate, declarative knowledge can have impact on behavior but that impact is filtered through an interpretive system which is well-oiled in achieving the goals of the system. This does not guarantee that new learning will not result in disaster but it does significantly lower the probability. If a new piece of knowledge proves to be faulty it can be tagged as such and so disregarded. It is much more difficult to correct a faulty procedure.

As a gross example, suppose I told a gullible child, "If you want something then you can assume it has happened." Translated into a production it would take on the following form

```
    IF the goal is to achieve X
  THEN POP with X achieved
```

This would lead to a perhaps blissful but deluded chld who never bothered to try to achieve anything because he believed it was already achieved. As a useful cognitive system he would come to an immediate halt. However, even if the child were gullible enough to encode this in declarative form at face value and perhaps even act upon it, he would quickly identify it as a lie (by contradiction procedures he has), tag it as such, and so prevent it from having further impact on behavior, and continue on a normal life of goal achievement. New information should enter in declarative form because one can encode information declaratively without committing control to it and because one can be circumspect about the behavioral implications of declarative knowledge.[1]

Knowledge Compilation

Interpreting knowledge in declarative form has the advantage of flexibility but it also has serious costs in terms of time and working memory space. The process is slow because the process of interpretation requires retrievals from long-term memory of declarative information and because the individual production steps of an interpreter are small in order to achieve generality. (For instance, the steps of problem refinement in Table 2 and Figure 5 were painfully small.) The interpretive productions require that the declarative information be represented in working memory and this can place a heavy burden on working memory capacity. Many subject errors and much of their slowness seem attributable to working memory errors. Students can be seen to repeat themselves over and over again as they lose critical intermediate results and have to recompute them.

The Phenomenon of Compilation

One of the processes in geometry that we have focussed on is how students match postulates against problem statements. Consider the side-angle-side (SAS) postulate whose presentation in the text is given in Figure 9. We followed a student through the exercises in the text that followed the section that contained this postulate and the side-side-side (SSS) postulate. The first problem that required use of SAS is illustrated in Figure 10. The following is the portion of his protocol where he actually called up this postulate and

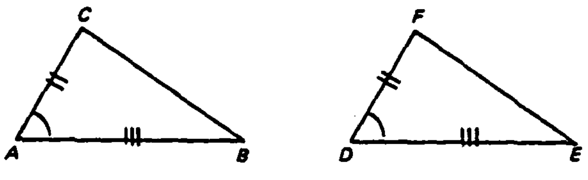
[1]As a side remark, I should acknowledge here that I am contradicting some of my earlier publications (e.g., Anderson, Kline, & Beasley, 1979, 1980) where I proposed a designation process that allowed productions to be directly created. This was rightfully criticized (e.g., Norman, 1980) as far too powerful computationally to be human. We were certainly always aware of the problems of designation--for instance in my discussion of induction in the 1976 ACT book (section 12.3), I was stubbornly avoiding such a process. However, a few years ago there seemed no way to construct a learning theory without such a mechanism. Now thanks to the development of ideas about knowledge compilation, the designation mechanism is no longer necessary.

managed to put it in correspondence to the problem:

> If you looked at the side-angle-side postulate--long pause--well RK and RJ could almost be--long pause--what the missing--long pause--the missing side. I think somehow the side-angle-side postulate works its way into here---long pause--Let's see what it says: "two sides and the included angle." What would I have to have to have two sides. JS and KS are one of them. Then you could go back to RS = RS. So that would bring up the side-angle-side postulate--long pause--But where would ∠1 and ∠2 are right angles fit in--long pause--wait I see how they work--long pause--JS is congruent to KS--long pause--and with angle 1 and angle 2 are right angles that's a little problem--long pause--OK, what does it say--check it one more time: "If two sides and the included angle of one triangle are congruent to the corresponding parts"--So I have got to find the two sides and the included angle. With the included angle you get angle 1 and angle 2. I suppose--long pause--they are both right angles which means they are congruent to each other. My first side is JS is to KS. And the next one is RS to RS. So these are the two sides. Yes, I think it is the side-angle-side postulate.

After reaching this point there was still a long process by which the student actually went through writing out the proof--but this is the relevant portion in terms of assessing what goes into recognizing the relevance of SAS.

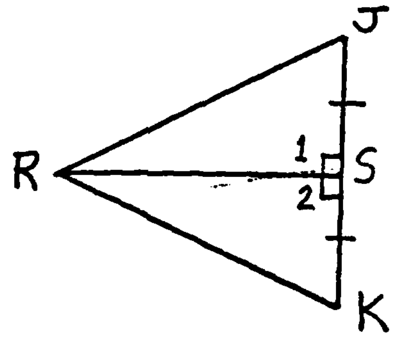
POSTULATE 14 (SAS POSTULATE) If two sides and the included angle of one triangle are congruent to the corresponding parts of another triangle, the triangles are congruent.



According to Postulate 14:

If $\overline{AB} \cong \overline{DE}$, $\overline{AC} \cong \overline{DF}$, and $\angle A \cong \angle D$, then $\triangle ABC \cong \triangle DEF$.

Figure 9: Statement in the text of the side-angle-side postulate.



Given: ∠1 and ∠2 are right angles
JS = KS
Prove: △RSJ = △RSK

Figure 10: The first proof generation problem that a student encounters which requires application of the SAS postulate.

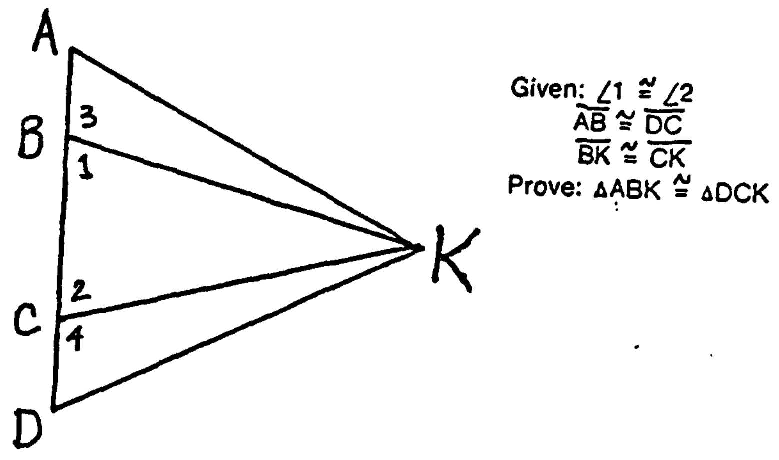


Figure 11: The fourth proof generation problem that a student encounters which requires application of the SAS postulate.

After a series of four more problems (two were solved by SAS and two by SSS), we came to the student's last application of the SAS postulate--for the problem illustrated in Figure 11. The method recognition portion of the protocol follows:

> Right off the top of my head I am going to take a guess at what I am supposed to do--∠DCK ≅ ∠ABK. There is only one of two and the side-angle-side postulate is what they are getting to.

A number of things seem striking about the contrast between these two protocols. One is, of course, there has been a clear speed-up in the application of the postulate. A second is that there is no verbal rehearsal of the statement of the postulate in the second case. We take this as evidence that the student is no longer calling a declarative representation of the problem into working memory. Note also in the first protocol that there are a number of failures of working memory--points where the student recomputed information that he has forgotten. The third feature of difference is that in the first protocol there is a clear piecemeal application of the postulate by which the student is separately identifying every element of the postulate. This is absent in the second protocol. It gives the appearance of the postulate being matched in a single step. These three features--speed-up, drop-out of verbal rehearsal, and elimination of piecemeal application--are among the features that we want to associate with the processes of knowledge compilation.

The Mechanisms of Compilation

The knowledge compilation processes in ACT can be divided into two subprocesses. One, which we call *composition*, takes sequences of productions that follow each other in solving a particular problem and collapses them into a single production that has the effect of the sequence. This produces considerable speed-up by creating new operators which embody the sequences of steps that are used in a particular problem domain. The second process, *proceduralization*, builds versions of the productions that no longer require the domain-specific declarative information to be retrieved into working memory. Rather the essential products of these retrieval operations are built into the new productions.

Most of this section is devoted to giving a rather detailed and technical analysis of compilation, but it is useful to have a less formal illustration first--and the reader can wade into the subsequent technical detail only if desired and, if so, with a better sense of its point. Consider the following two productions that might serve to dial a telephone number.

```
    IF the goal is to dial a telephone number
       and digit1 is the first digit of the number
THEN dial digit1

    IF the goal is to dial a telephone number
       and digit1 has just been dialed
       and digit2 is after digit1 in the number
THEN dial digit2
```

Composition can create "macro-production" which does the operation of this sequence of two productions. So from this pair we might create

```
    IF the goal is to dial a telephone number
       and digit1 is the first digit of the number
       and digit2 is after digit1
THEN dial digit1 and then digit2
```

Compositions like this will reduce the number of production applications to perform the task. Such a production still requires that the phone number be held in working memory. It is possible to eliminate this requirement by building special productions for special numbers. This is the function of proceduralization. So, proceduralization applied to Mary's number (432-2815) and the above production, would produce:

```
    IF the goal is to dial Mary's number
THEN dial 4 and then 3
```

By continued composition and proceduralization a production can be built that dials the full number:

```
    IF the goal is to dial Mary's number
THEN dial "432-2815"
```

This, by the way, corresponds to the experience of some, including myself (Anderson, 1976, 1980), of knowing certain phone numbers in terms of procedure for dialing them rather than in terms of a declarative fact.

Encoding and Application of the SAS Postulate

The details of these mechanisms of knowledge compilation are best explained in the context of an example. The example traces the evolution of the SAS postulate from a declarative state to a procedural state.

Table 3
Encoding of the SAS Postulate

SAS-Background
- S1 is a side of ΔXYZ
- S2 is a side of ΔXYZ
- A1 is an angle of ΔXYZ
- A1 is included by S1 and S2
- S3 is a side of ΔUVW
- S4 is a side of ΔUVW
- A2 is an angle of ΔUVW
- A2 is included by S3 and S4

SAS-Hypotheses
- S1 is congruent to S3
- S2 is congruent to S4
- A1 is congruent to A2

SAS-Conclusion
- ΔXYZ is congruent to ΔUVW

Table 3 provides a schema-like encoding of the SAS postulate. This schema is just a set of facts that encodes the critical information in the side-angle-side postulate. It is segmented into a set of propositions about the background, a set of propositions which provide the hypotheses of the postulate, and a conclusion. The background serves to provide a description of the relevant aspects of the diagram, particularly identifying the relevant elements like S1, S2, A1 which serve as the variables of the representation. Subjects spend more time trying to relate the postulate to the diagram than anything else--consistent with the number of propositions in the background. This is a highly structured encoding of the postulate and the structure is critical to correct use of the postulate. We have examples of student failure attributable to incorrect structuring of postulate encoding (e.g., getting antecedent and consequent confused).

Table 4 provides some of the first productions that might apply in an interpretive attempt to use this postulate in reasoning backwards. Figure 12 illustrates their flow of control. In order to be able to trace out the application of knowledge compilation I have had to make explicit their variable structure in Table 4. Suppose the system has the goal to prove ΔABC = ΔDEF. I want to go very carefully through the first three productions to apply in this case because I will be using them to explain composition and proceduralization. Production P1 is evoked with the following assignment of clauses:

The goal is to prove that LVobject1 has LVrelation to LVobject2
--> The goal is to prove that ΔABC is congruent to ΔDEF.

LVschema has as conclusion that LVobject3 has LVrelation to LVobject4
--> SAS-schema has as conclusion that ΔXYZ is congruent to ΔUVW

LVschema has LVbackground as background
--> SAS-schema has SAS-background as background

Table 4
Some Productions Used in the Backward Application of the Schema in Table 3

```
P1:       IF the goal is to prove that LVobject1 has LVrelation to LVobject2
             and LVschema has as conclusion that LVobject3 has LVrelation to LVobject4
             and LVschema has LVbackground as background
             and LVschema has LVhypotheses as hypotheses
        THEN LVobject1 corresponds to LVobject3
             and LVobject2 corresponds to LVobject4
             and set as subgoals to match LVbackground
                and to prove LVhypotheses

P2:       IF the goal is to match LVbackground
             and LVbackground begins with LVstatement
        THEN set as a subgoal to match LVstatement

P3:       IF the goal is to match LVbackground
             and LVstatement1 has just been matched
             and LVstatement2 follows LVstatement1
        THEN set as a subgoal to match LVstatement2

P4:       IF the goal is to match that LVobject1 has LVrelation to LVobject2
             and LVobject3 corresponds to LVobject2
             and the problem has given that LVobject4 has LVrelation to LVobject3
        THEN LVobject4 corresponds to LVobject1
             and POP the goal

P5:       IF the goal is to match that LVobject1 has LVrelation to LVobject2 and LVobject3
             and LVobject4 corresponds to LVobject1
             and LVobject5 corresponds to LVobject2
             and LVobject6 corresponds to LVobject3
             and the problem has given that LVobject4 had LVrelation to LVobject5 and LVobject6
        THEN POP the goal

P6:       IF the goal is to match the LVbackground
             and the last statement has been matched
        THEN POP the goal
```

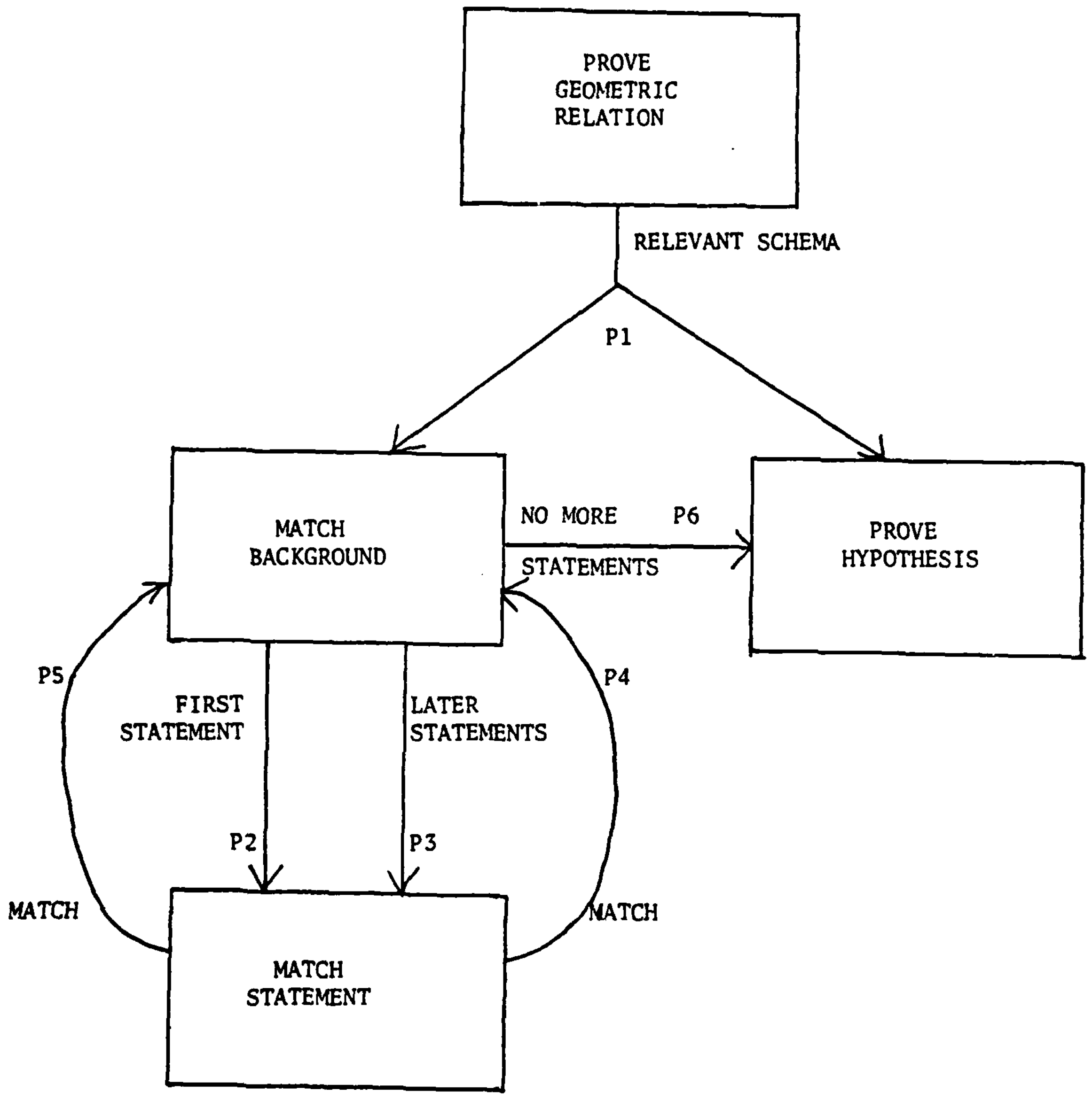


Figure 12: A representation of the flow of control in Table 4 between the various goals.

LVschema has LVhypotheses as hypotheses
--> SAS-schema has SAS-hypotheses as hypotheses

The first clause matches against the goal in working memory and the remaining three against elements of the postulate schema in long term memory. The term SAS-background and SAS-hypotheses in the above refer to nodes that organize the background and hypotheses clauses. The critical feature which enabes P1 to appropriately select the SAS postulate is that it tests that the conclusion of the schema establishes the same relation as the relation in the goal. It tests for this by the use of the same local variable, LVrelation, in both the first and second condition clauses. P1 in its action adds the following information to working memory.

ΔABC corresponds to ΔXYZ

ΔDEF corresponds to ΔUVW

. These correspondences are put in working memory to aid subsequent productions in matching the schema background to the problem. A constraint in performing this match is that these correspondences be kept. P1 also sets subgoals to match the background of the SAS-schema and then to prove the hypotheses.

P2 is then evoked. Its conditions are matched as follows:

The goal is to match LVbackground
--> The goal is to match SAS-background

LVbackground begins with LVstatement
--> SAS-background begins with "S1 is a side of ΔXYZ"

Note in matching the second clause, P2 retrieves from memory the first statement in the SAS background. In its action it sets as its goal to match this statement.

Next to apply is P4. Its conditions are matched as follows:

The goal is to match that LVobject1 has LVrelation to LVobject2
--> The goal is to match that S1 is side of ΔXYZ

LVobject3 corresponds to LVobject2
--> ΔABC corresponds to ΔXYZ

The problem has given that LVobject4 has LVrelation to LVobject1
--> The problem has given that $\overline{AB}$ is a side of ΔABC.

This production thus determines that "$\overline{AB}$ is a side of ΔABC" in the problem can be matched to "S1 is a side of ΔXYZ" in the schema and POPs. It also puts into working memory the correspondence:

$\overline{AB}$ corresponds to S1

After this point productions P3 and P4 or P5 will repeat in cycle matching all the statements in the background. Then production P6 will apply and POP the goal of matching the background. This will turn the system to the goal of trying to prove the hypotheses. A general matching system would need more than P4 and P5 and in Neves and Anderson (1981) we offer a more general solution for interpretive pattern-matching, but the above will do for current purposes.

It should be clear that this is quite a general set of productions--indeed only production P1 is specific to inference schemata or their backgrounds and it still has a very broad range of applicability. It applies to the side-angle-side postulate only as a very special case.

Composition

Composition works by building productions that capitalize on the regularities in the sequence of production application in a particular domain. The basic idea is to build single productions which have the effect of sequences of productions. The idea of collapsing multiple cognitive steps into a single step can at least be traced back to Book's (1908) analysis of the acquisition of typewriting skills. A modern-day rendition of it in terms of production systems is provided by Lewis (1979). The process of knowledge compilation can be particularly well exploited in the ACT production system architecture.

Composition takes two productions *IF A THEN B* and *IF C THEN D* that have applied in sequence and builds a new production *IF A&(C-B) THEN B&D*. *A&(C-B)* denotes the union of those conditions in *A* and those in *C* not provided by action *B*. *B&D* just denotes the union of actions *B* and *D*. Let us illustrate this with respect to the composition of P1 and P2 in this example. Their composition is:

```
P1&P2:     IF the goal is to prove that LVobject1 has LVrelation to LVobject2
              and LVschema has as conclusion that LVobject3 has LVrelation to LVobject4
              and LVschema has LVbackground as background
              and LVschema has LVhypotheses as hypotheses
              and LVbackground begins with LVstatement
         THEN LVobject1 correresponds to LVobject3
              andLVobject2 corresponds to LVobject4
              and set as subgoals to match LVstatement
                 and to match LVbackground
                 and to prove LVhypotheses
```

All the conditions of P1 are used and all the conditions of P2 except "the goal is to match LVbackground" which was provided in the action of P1. The action of P1&P2 includes all the actions of P1 and P2. This production now can do in a single step what P1 and P2 did in two.

Suppose this production P1&P2 is composed with P4 that follows. The result is:

```
P1&P2&P4:  IF the goal is to prove LVobject1 has LVrelation to LVobject2
              and LVschema has as conclusion that LVobject3 has LVrelation to LVobject4
              and LVschema has LVbackground as background
              and LVschema has LVhypotheses as hypotheses
              and LVbackground begins with "LVobject5 has LVrelation1 to LVobject3"
              and the problem has given that LVobject6 has LVrelation1 to LVobject1
```

```
THEN LVobject1 corresponds to LVobject3
     and LVobject2 corresponds to LVobject4
     and LVobject6 corresponds to LVobject5
     and set as subgoals to match LVbackground
        and to prove LVhypotheses
```

The condition of this production contains all the clauses from the condition of P1&P2 plus the third clause from P4. The first and second clauses of P4 are omitted because they were provided in the action of prior P1&P2.

There are some additional interesting complications here. Statement *LVbackground begins with LVstatement* in P1&P2 has been expanded into *LVbackground begins with "LVobject5 has LVrelation1 to LVobject1"*. This unpacking of LVstatement was used in P4 and composition always uses the more specific designation of a structure in the two productions it is composing. Also note that composition determines that the LVobject1 in P1&P2 is the same as LVobject3 in P4 and it uses the common LVobject1 for both in P1&P2&P4. Similarly it uses the common LVobject3 for LVobject3 from P1&P2 and LVobject2 from P4. Thus, composition encodes correspondences in variable bindings that the two productions had in this sequence. Thus, the composed production is more specific in its constraints than the original P1, P2, and P4.

Also note that in its action P1&P2 sets the goal to match the statement and in its action P4 popped this goal. The setting and popping of this goal is simply omitted in P1&P2&P4. This is one example where a general learning mechanism can take advantage of this semantics of goal structures to simplify the productions it produces.

To review, the basic function of composition is to collapse into single steps productions which in general may apply independently (i.e., they do not always follow one upon the other). Composition capitalizes on the fact that they do follow each other on a specific knowledge application. It also capitalizes on features of that application to reduce number of clauses and variables. So while the original three productions involve 16 clauses (conditions and actions) and 15 variables P1&P2&P4 involves 11 clauses and 11 variables.

Remarks about the Composition Mechanism. In the above discussion and in the computer implementation the assumption has been that a pair of productions will be composed if they follow each other. This means that upon repeated applications of the same problem, the number of productions should be halved each time. More generally, however, one might assume that the number of productions in each application is reduced to a proportion a of the previous application that involved composition. If $a > 1/2$ this might reflect that compositions are formed with probability less than 1. If $a < 1/2$ this might reflect the fact that composition involved more than a pair of productions. Thus after n compositions the expected number of productions would be Na^n where N was the initial number. As will be argued later, the rate of composition (n) may not be linear in number of applications of the production set to problems.

There is a limitation on how large production conditions can get and so a limitation on composition. A production will only match if there is in working memory propositions that correspond to each clause in the production's condition. If a production is created whose condition exceeds the size of working memory it will not apply and so cannot enter into further compositions. However, as we will discuss later, it may be possible to increase the capacity of working memory with practice.

There is the opportunity for spurious pairs of productions to accidently follow each other and so be composed together. If we allowed spurious pairs of productions to be composed together there would not be disastrous consequences but it would be quite wasteful. Also, spurious productions might intervene between the application of productions that really belong together. So, for instance, suppose the following three productions had happened to apply in sequence:

```
P1:         IF the subgoal is to add in a digit
          THEN set as a subgoal to add the digit and the running total

P2:         IF I hear footsteps in the aisle
          THEN the teacher is coming my way

P3:         IF the goal is to add two digits
               and a sum is the sum of the two digits
          THEN the result is the sum
               and POP
```

This sequence of productions might apply, for instance, as a child is performing arithmetic exercises in a class. The first and third are set to process subgoals in the solving of the problem. The first sets up the subgoal that is met by the third. The second production is not related to the other two and is merely an inference production that interprets sounds of the teacher approaching. It just happens to intervene between the other two. Composition as described would produce the following pairs:

```
P1&P2:      IF the subgoal is to add in a digit
               and I hear footsteps in the aisle
          THEN set as a subgoal to add the digit and the running total
               and the teacher is coming my way

P2&P3:      IF I hear footsteps in the aisle
               and the goal is to add two digits
               and a sum is the sum of the two digits
          THEN the teacher is coming my way
               and the result is the sum
               and POP
```

These productions are harmless but basically useless. They have also prevented formation of the following, useful composition:

```
P1&P3:      IF the subgoal is to add in a digit
               and the sum is the sum of the digit and the running total
```

THEN the result is the sum

Therefore, it seems reasonable to advance a sophistication over the composition mechanism proposed in Neves and Anderson. In this new scheme productions are composed only if they are linked by goal setting (as in the case of P1 & P3) and productions that are linked by goal setting will be composed even if intervening there are productions which make no goal reference (as in the case of P2). This is another example where the learning mechanisms can profitably exploit the goal-structuring of production systems.

Proceduralization

As noted above, one factor limiting the formation of composition is that productions with larger conditions require more information to be held in working memory. Proceduralization has as one of its motivations that it reduces the demand on working memory for production execution. Specifically, proceduralization eliminates the need for long-term memory information to be retrieved into working memory for matching by a production's condition. Rather, the products of the long-term memory retrieval are directly built into the production.

Let us consider how the proceduralization would apply to the composed production P1&P2&P4. Note that the second, third, fourth, and fifth clauses of this production's conditions all match to information in long-term memory encoding the SAS postulate. The effect of matching these four clauses to long-term memory is to constrain the values of certain local variables and hence to constrain the matching of other condition clauses and to determine in part what the action clauses achieve. In the example where P1&P2&P4 matches to the SAS schema we get the following bindings of variables:

LVschema = SAS-schema
LVobject3 = ∆XYZ
LVrelation = is congruent to
LVobject4 = ∆UVW
LVbackground = SAS-background
LVhypotheses = SAS-hypotheses
LVobject5 = S1
LVrelation1 = is side of

The long-term memory propositions matched to the clauses are always true and so the only reason for matching them in the condition of P1&P2&P4 is to achieve these variable bindings. We can get the effect of matching these long-term memory facts simply by constraining these variables to have these values elsewhere in the production. This can be done by replacing these variables by their values. If we do this, we can then delete matches to these long-term memory propositions from the condition. The following production results:

```
P1&P2&P4':        IF the goal is to prove LVobject1 is congruent to LVobject2
                     and the problem has given that "LVobject6 is a side of LVobject1"
                THEN LVobject1 corresponds to ∆XYZ
                     and LVobject2 corresponds to ∆UVW
                     and LVobject6 corresponds to S1
```

```
        and the subgoals are to match SAS-background
            and to prove SAS-hypotheses
```

The above production has reduced the condition side to 2/6 of the original production and greatly reduced the number of variables. Moreover, memory for the second condition is supported by the external diagram. So it is possible that productions composed from this will be able to apply whereas productions composed from the original P1&P2&P4 could not because of excess demands for the maintenance of information in working memory. As with composition, our implementation of proceduralization has it applying all the time, but again it might be more reasonable to propose that proceduralization was a probabilistic affair. In this case proceduralization and composition would be closely interlocked such that the probability of a composition increased when a satisfactory proceduralization occurs.

Further Composition and Proceduralization. It is interesting to inquire what would happen if composition and proceduralization continued in the productions P1 through P6 until there was a single, proceduralized production to match the total background of the schema. The final product would be:

```
  IF the goal is to prove LVobject1 is congruent to LVobject2
     and LVobject3 is a side of LVobject1
     and LVobject4 is a side of LVobject1
     and LVobject5 is an angle of LVobject1
     and LVobject5 is included by LVobject3 and LVobject4
     and LVobject6 is a side of LVobject2
     and LVobject7 is a side of LVobject2
     and LVobject8 is an angle of LVobject2
     and LVobject8 is included by LVobject6 and LVobject7
THEN LVobject1 corresponds to ΔXYZ
     and LVobject2 corresponds to ΔUVW
     and LVobject3 corresponds to S1
     and LVobject4 corresponds to S2
     and LVobject5 corresponds to A1
     and LVobject6 corresponds to S3
     and LVobject7 corresponds to S4
     and LVobject8 corresponds to A2
     and the subgoal is to prove SAS-hypotheses
```

This production matches in one step the whole background of the postulate and in its action sets the goal to prove the hypotheses. It also records all the correspondences between parts of the schema and elements of the diagram. This information will be used in deciding what to prove congruent to what. Often in later portions of this paper, we will refer to such a production as

```
  IF the goal is to prove $\Delta XYZ \cong \Delta UVW$
THEN set as subgoals to prove
          1. $\overline{XY} \cong \overline{UV}$
          2. $\overline{YZ} \cong \overline{VW}$
          3. $\angle XYZ \cong \angle UVW$
```

for shorthand. However, in actual implementation the shorthand would need an expansion more like the first rendition of this production. This expanded rendition also makes the point that, while the efficiencies in composition and proceduralization do much to reduce the size of productions, there is an inevitable increase in both condition size and action size with composition. Thus, limits on the capacity of working memory will still put limits on the scope of composition.

The impact of composition and proceduralization is to create domain-specific procedures. Thus, in combination they transform the performance of the skill from interpretative application of declarative knowledge to direct application of procedural knowledge.

Evidence for Knowledge Compilation

It is worth reviewing the kind of evidence that indicates knowledge compilation goes on. We have already emphasized the rapid initial speed-up and this need not be mentioned further--but we will return to the issue of the form of the speed-up later. We have also noted that there is a loss of verbal mediation with practice. This is produced by a diminishing need to rehearse the material as the knowledge becomes more proceduralized. I would like to consider in detail here two other phenomena--the disappearance of effects of memory size and display size in the scan task and the Einstellung effect in problem-solving.

The Sternberg Paradigm. In the Sternberg paradigm (e.g. Sternberg, 1969) subjects are asked to indicate if a probe comes from a small set of items. The classic result is that decision time increases with set size. It has been shown that effects of size of memory set can diminish with repeated practice (Briggs & Blaha, 1969). A sufficient condition for this to occur is that the same memory set be used repeatedly. The following are two productions that a subject might use for performing the scan task at the beginning of the experiment:

```
PA:        IF the goal is to recognize LVprobe
              and LVprobe is a LVtype
              and the memory set contains a LVitem of LVtype
         THEN say YES
              and POP the goal

PB:        IF the goal is to recognize LVprobe
              and LVprobe is a LVtype
              and the memory set does not contain a LVitem of LVtype
         THEN say NO
              and POP the goal
```

In the above, LVprobe and LVitem will match to tokens of letters and LVtype match to a particular letter type (e.g., the letter A). This production set is basically the same as the production system for the Sternberg task given in Anderson (1976) except in a somewhat more readable form that will expose the essential character of the processing. These productions require that the contents of the memory set be held active in working memory. As discussed in Anderson (1976), the more items required to be held active in working memory the lower the activation of each and the slower the recognition judgment--which produces the typical set size effect.

Consider what happens when these productions apply repeatedly in the same list--say a list consisting of A, J, and N with foils coming from a list of L, B, K. Then through proceduralization we would get the following productions from PA:

```
P1:        IF the goal is to recognize LVprobe
              and LVprobe is an A
        THEN say YES
              and POP the goal

P2:        IF the goal is to recognize LVprobe
              and LVprobe is a J
        THEN say YES
              and POP the goal

P3:        IF the goal is to recognize LVprobe
              and LVprobe is a N
        THEN say YES
              and POP the goal
```

The preceding are productions for recognizing the positive set. Specific productions would also be produced by proceduralization from PB to reject the foils:

```
P4:        IF the goal is to recognize. LVprove
              and LVprobe is a L
        THEN say NO
              and POP the goal

P5:        IF the goal is to recognize LVprobe
              and LVprobe is a B
        THEN say NO
              and POP the goal

P6:        IF the goal is to recognize LVprobe
              and LVprobe is a K
        THEN say NO
              and POP the goal
```

It is interesting to note here that Shiffrin and Dumais (1981) report that the automatization effect they observe in such tasks is as much due to subjects' ability to reject specific foils as it is their ability to accept specific targets. These productions no longer require the memory set to be held in working memory and will apply in a time independent of memory set size. However, there still may be some effect of set size in the subject's behavior. These productions do not replace PA and PB; rather, they coexist and it is possible for a classification to proceed by the original PA and PB. Thus, we have two parallel bases for classification racing, with the judgment being determined by the fastest. This will produce a set size effect which will diminish as P1 - P6 become strengthened.

The Scan Task. Shiffrin and Schneider (1977) report an experiment in which they gave subjects a set of items to remember. Then subjects were shown in rapid succession a series of displays where each display contained a set of items. Subjects' task was to decide if any of the displays contained an item in the memory set. When Shiffrin and Schneider kept the members of the study set constant and the distractors constant, they found considerable improvement with practice in subjects' performance on the task. They interpreted their results as indicating both a diminishing effect of memory set size and of the number of alternatives in the display. Consider what a production set might be like that scanned an array to see if any member of the array matched a memory set item:

```
PC*         IF the goal is to see if LVarray contains a memory item
                and LVprobe is in POSITION*
            THEN the subgoal is to recognize LVprobe

PD:         IF the goal is to recognize LVprobe
                and LVprobe is a LVtype
                and the memory set contains LVitem of LVtype
            THEN tag the goal as successful
                and POP the goal

PE:         IF the goal is to recognize LVprobe
                and LVprobe is a LVtype
                and the memory set does not contain a LVitem of LVtype
            THEN tag the goal as failed
                and POP the goal

PF:         IF the goal is to see if LVarray contains a memory item
                and there is a successful subgoal
            THEN say YES
                and POP the goal

PG:         IF the goal is to see if LVarray contains a memory item
            THEN say NO
                and POP the goal
```

Production PC* is a schema for a set of productions such that each one would recognize an item in a particular position. An example might be

```
            IF the goal is to see if LVarray contains a memory item
                and LVprobe is in the upper-right corner
            THEN set as a subgoal to recognize LVprobe
```

PD and PE are similar to PA and PB given earlier--they check whether each position focused by a PC* contains a match. PF will apply if one of the probes lead to a successful match and the default production PG will apply if none of the positions leads to success. The behavior of this production set is one in which *individual versions of the* PC* apply serially, focusing attention on individual positions. PD and PE are responsible for the judgment of individual probes. This continues until a positive probe is hit and PF applies or until there are no more probe positions and PG applies. (PG will only be selected when there are no more positions because specificity will prefer PC* and PF over it.) Because of the need to keep the memory set active, an effect of set size is expected. The serial examination of positions produces an effect of display size. These two factors should be multiplicative which is what Schneider and Shiffrin (1977) report.

Consider what will happen with knowledge compilation. Composing a PC* production with PD and with PF and proceduralizing, we will get positive productions of the form:

```
P7:        IF the goal is to see if LVarray contains a memory item
              and the upper right hand position contains a LVprobe
              and the LVprobe is an A
        THEN say YES
              and POP the goal
```

The negative production would be formed by composing together a sequence of PC* productions paired with PE and a final application of PG. All the subgoal setting and popping would be composed out. The strict composition of this sequence would be productions like:

```
P6:        IF the goal is to see if LVarray contains a memory item
              and the upper left hand position contains a LVprobe1
              and LVprobe1 is a K
              and the upper right hand position contains a LVprobe2
              and LVprobe2 is a B
              and the lower left hand position contains a LVprobe3
              and LVprobe3 is a L
              and the lower right hand position contains a LVprobe4
              and LVprobe is a K
        THEN say NO
              and POP the goal
```

where a separate such production would have to be formed for each possible foil combination. These productions predict no effect of set size or probe size which is consistent with the Schneider and Shiffrin findings.

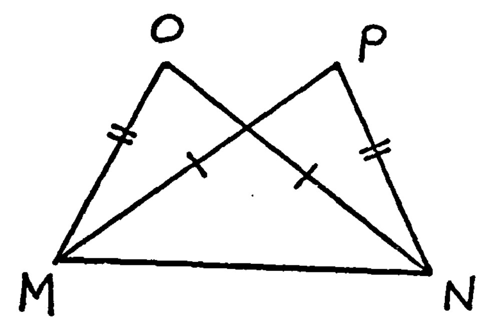
The Einstellung Phenomenon. Another phenomena attributable to knowledge compilation is the Einstellung effect (Luchins, 1942; Luchins & Luchins, 1959) in problem-solving. One of the types of examples used by Luchins to demonstrate this phenomena is illustrated in Figure 13. Luchins presented his subjects with a sequence of geometry problems like the one in Part (a). For each problem in the sequence the student had to prove two triangles congruent in order to prove two angles congruent. Then subjects were given a problem like the one in Part (b). Subjects proved this by means of congruent triangles even though it has a much simpler proof by means of vertical angles. Subjects not given the initial experience with problems like the one in Part (a) show a much greater tendency to use the vertical angle proof. Their experimental experience caused subjects to solve the problem in a non-optimal way.

Lewis (1978) has examined the Einstellung effect and its relation to the composition process. He defines as *perfect composites* compositions that do not change the behavior of the system but just speed it up. Such compositions cannot produce Einstellung, of course. However, he notes that there are a number of natural ways to produce non-perfect composites that produce Einstellung. The ACT theory provides an example of such a non-perfect composition process. Composites are non-perfect in ACT because of its conflict resolution principles.

Production P1 through P4 provide a model of part of the initial state of the student's production system.

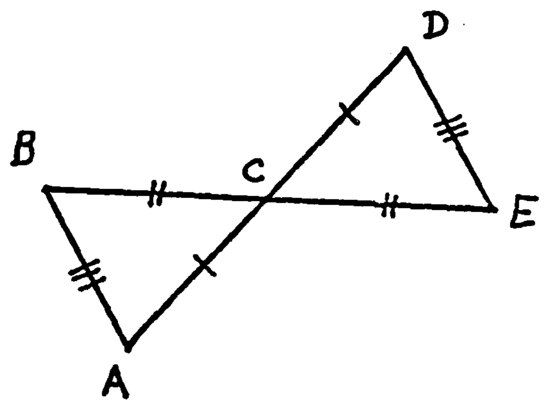
Earlier Examples Like:

(a)



Given: $\overline{OM} \cong \overline{PN}$
$\overline{MP} \cong \overline{NO}$
Prove: $\angle MON \cong \angle NPM$

(b)



Given: $\overline{AC} \cong \overline{CD}$
$\overline{BC} \cong \overline{CE}$
$\overline{AB} \cong \overline{DE}$
Prove: $\angle BCA \cong \angle DCE$

Figure 13: After solving a series of problems like (a) students are more likely to choose the non-optimal solution for (b).

P1: IF the goal is to prove $\angle XYZ \cong \angle UVW$
and the points are ordered X, Y, and W on a line
and the points are ordered Z, Y, and U on a line
THEN this can be achieved by vertical angles
and POP the goal

P2: IF the goal is to prove $\angle XYZ \cong \angle UVW$
THEN set a subgoals
1. To find a triangle that contains $\angle XYZ$
2. To find a triangle that contains $\angle UVW$
3. To prove the two triangles congruent
4. To use corresponding parts of congruent triangles

P3: IF the goal is to find a figure that has a relation to an object
and Figure X has the relation to the object

```
        THEN the result is Figure X
                and POP the goal

P4:         IF the goal is to prove $\Delta XYZ \cong \Delta UVW$
                and $\overline{XY} \cong \overline{UV}$
                and $\overline{YZ} \cong \overline{VW}$
                and $\overline{ZX} \cong \overline{WU}$
        THEN this can be achieved by SSS
                and POP the goal
```

Production P1 is responsible for immediately recognizing the applicability of the vertical angle postulate. Productions P2 - P4 are part of the production set that is responsible for proof through the route of corresponding parts of congruent triangles. Production P2 decomposes the main goal into the subgoals of finding the containing triangles, of proving they are congruent, and then of using the corresponding parts principle. P3 finds the containing triangles, and P4 encodes one production that would recognize triangle congruence. This production set, applied to a problem like that in Part (b) of Figure 13, would lead to a solution by vertical angles. This is because production P1, for vertical angles, is more *specific* in its condition than production P2 which starts off the corresponding angles proof. As explained earlier, ACT's conflict resolution prefers specific productions.

Consider, however, what would happen after productions P2 - P4 had been exercised on a number of problems and composition had taken place. Production P2&P3&P3&P4 represents a composition of the sequence P2, then P3, then P3, and then P4. Its condition is not less specific than P1 and, in fact, contains more clauses. However, because these clauses are not a superset of P1's clauses, it is not the case that either production is technically more specific than the other. They are both in potential conflict and, because both change the goal state, application of one will block the application of the other. In this case, strength serves as the basis for resolving the conflict. Production P2&P3&P3&P4, because of its recent practice, may be stronger and therefore would be selected.

```
P2&P3&P3&P4:
            IF the goal is to prove $\angle XYZ \cong \angle UVW$
                and $\angle XYZ$ is part of $\Delta XYZ$
                and $\angle UVW$ is part of $\Delta UVW$
                and $\overline{XY} \cong \overline{UV}$
                and $\overline{YZ} \cong \overline{VW}$
                and $\overline{ZX} \cong \overline{WU}$
        THEN set $\Delta XYZ \cong \Delta UVW$
                and set as a subgoal to use corresponding part of congruent triangles
```

This example illustrates how practice through composition can change the specificity ordering of productions and how it can directly change the strength. These two factors, change of specificity and change of strength, can cause ACT's conflict resolution mechanism to change the behavior of the system, producing Einstellung. Under this analysis it can be seen that Einstellung is an aberrant phenomenon reflecting what is basically an adaptive adjustment on the system's part. Through strength and composition ACT is unitizing and favoring sequences of problem-solving behaviors that have been successful recently. It is a good bet that such sequences will prove useful again. It is to the credit of the cleverness of Luchins' design that it exposed the potential cost of these usually beneficial adaptations.

It has been suggested that one could produce the Einstellung effect by simply strengthening particular productions. So one might suppose that production P2 is strengthened over P1. The problem with this explanation is that subjects can be shown to have a preference for a particular *sequence* of productions not single productions in isolation. Thus, in the waterjug problems, described by Luchins, subjects will fixate on a specific sequence of operators implementing a subtraction method and will not notice other simpler subtraction methods. The composition mechanism explains how the subject encodes this operator sequence.

It is interesting to compare the time scale for producing Einstellung with the time scale for producing the automatization effects in the Sternberg paradigm and the scan paradigm. Strong Einstellung effects can be produced after a half dozen trials whereas the automatization results will require hundreds of trials. This suggests that composition which underlies Einstellung can proceed more rapidly than the proceduralization which underlies the automatization effects. Proceduralization is really more responsible for creating domain-specific procedures than is composition. Composition creates productions that encode the sequence of general productions for a task but the composed productions are still general. In contrast, by replacing variables with domain constants, proceduralization creates productions that are committed to a particular task. Apparently, the learning system is reluctant to create this degree of specialization unless there is ample evidence that the task will be repeated frequently.

The Adaptive Value of Knowledge Compilation

In the previous section on initial encoding it was argued that it was dangerous for a system to directly create productions to embody knowledge. For this reason and for a number of others it was argued that knowledge should first be encoded declaratively and then interpreted. This declarative knowledge could affect behavior but only indirectly via the intercession of existing procedures for correctly interpreting that knowledge. We have in the processes of composition and proceduralization a means of converting declarative facts into production form.

It is important to note that productions created from compilation really do not change the behavior of the system, except in terms of possible reorderings of specificity relations as noted in our discussion of Einstellung. Thus, knowledge compiled in this way has much of the same safeguards built into it that interpretative application of the knowledge does. The safety in interpretative applications is that a particular piece of knowledge does not impact upon behavior *until* it has undergone the scrutiny of all the system's procedures (which can, for instance, detect contradiction of facts or of goals). Because compilation only operates on successful sequences of productions that pass this scrutiny, it tends to only produce production embodiments of knowledge that pass that scrutiny. This is the advantage of learning from doing. Another advantage with interpretive application is that the use of the knowledge is forced to be consistent with existing conventions for passing control among goals. By compiling from actual use of this knowledge, the compiled productions are guaranteed to be likewise consistent with the system's goal structure.

We can understand why human compilation is gradual (in contrast to computer compilation) and occurs as a result of practice if we consider the difference between the human situation and the typical computer situation. For one thing, the human does not know what is going to be procedural in an instruction until he tries to use the knowledge in the instruction. In contrast, the computer has this built in, in the difference between program and data. Another reason for gradual compilation is to provide some protection against the errors that enter into a compiled procedure because of the omission of conditional tests. For instance, if the system is interpreting a series of steps that include pulling a lever, it can first reflect on the lever-pulling step to see if it involves any unwanted consequences in the current situation. These tests will be in the form of productions checking for error conditions. (These error-checking productions can be made more specific so that they would take precedence over the normal course of action.) When that procedure is totally compiled, the lever-pulling will be part of a pre-packaged sequence of actions with many conditional tests eliminated (see the discussion of Einstellung). If the procedure transits gradually between the interpretive and compiled stages, it is possible to detect the erroneous compiling out of a test at a stage where the behavior is still being partially monitored interpretively and can be corrected. It is interesting to note here the folk wisdom that most errors in acquisition of a skill, like airplane flying, occur neither with the novices nor with experts. Rather, they occur at intermediate stages of development. This is presumably where the conversion from procedural to declarative is occurring and the point where unmonitored mistakes might slip into the performance. So by making compilation gradual one does not eliminate the possibility of error, but one does reduce the probability.

Procedural Learning: Tuning

There is much learning that goes on after the skill has been compiled into a task-specific procedure and this learning cannot be just attributed to further speed-up due to more composition. One type of learning involves an improvement in the choice of method by which the task is performed. All tasks can be characterized as having a search associated with them athough in some cases the search is trivial. By search I mean that there are alternate paths of steps by which the problem can be tackled and the subject must choose between them. Some of these paths lead to no solution and some lead to more complex solutions than necessary. A clear implication of much of the novice-expert research (e.g., Larkin, McDermott, Simon, & Simon, 1980) is that what happens with high levels of expertise in a task domain is that the problem-solver becomes much more judicious in his choice of paths and may fundamentally alter his method of search. In terms of the traditional learning terminology, the issue is similar to, though by no means identical to, the issue of trial and error versus insight in problem solving. A novice's search of a problem space is largely a matter of trial and error exploration. With experience the search becomes more selective and more likely to lead to rapid success. I refer to the learning underlying this selectivity as *tuning*. My use of the term is quite close to that of Rumelhart and Norman (1976).

In 1977 we (Anderson, Kline, & Beasley, 1977) proposed a set of three learning mechanisms which still serve as the basis for much of our work on the tuning of search. There was a generalization process by which production rules became broader in their range of applicability, a discrimination process by which the rules became narrower, and a strengthening process by which better rules were strengthened and poorer rules weakened. These ideas have not-accidental relationships to concepts in the traditional learning literature, but as we will see they have been somewhat modified to be computationally more adequate. One can think of production rules as implementing a search where individual rules correspond to individual operators for expanding the search space. Generalization and discrimination serve to produce a "meta-search" over the production rules looking for the right features to constrain the application of these productions. Strength serves as an evaluation for the various constraints produced by the other two processes.

This section will consider tuning from two different perspectives. First, I will illustrate how these three central learning constructs operate in the ACT system with language acquisition examples. These learning mechanisms were originally conceived with respect to language processing and later extended to other problem-solving domains. It is a major claim of the theory that these learning mechanisms will apply equally well to domains as diverse as language processing and geometry proof generation. Second, after having described the mechanisms I show how they can produce tuning in a problem-solving domain like geometry. As part of this consideration of problem-solving, I will discuss also how we have applied composition (already discussed with respect to knowledge compilation) to produce tuning. We have not systematically developed the application of composition to language processing, although I think it can be done profitably.

Generalization

The ability to perform successfully in novel situations is the hallmark of human cognition. For example, *productivity* has often been identified as the most important feature of natural languages, where this refers to the speaker's ability to generate and comprehend utterances never before encountered. Traditional learning theories have been criticized because of their inability to account for this productivity (e.g., McNeill, 1968), and it was one of our goals in designing ACT to avoid this sort of criticism.

An Example. ACT's generalization algorithm looks for commonalities between a pair of productions and creates a new production rule which captures what these individual production rules have in common. As an example, consider the following pair of rules for language generation which might arise as the consequence of compiling productions to encode specific instances of phrases:

P1: IF the goal is to indicate that a coat belongs to me
THEN say "My coat"

```
P2:        IF the goal is to indicate that a ball belongs to me
       THEN say "My ball"
```

From these two production rules ACT can form the following generalization:

```
P3:        IF the goal is to indicate that LVobject belongs to me
       THEN say "My LVobject"
```

in which the variable LVobject has replaced the particular object.[2] The rule now formed is productive in the sense that it will fill in the LVobject slot with any object. Of course, it is just this productivity in child speech which has been commented upon at least since Braine (1963). It is important to note that the general production does not replace the original two and that the original two will continue to apply in their special circumstances.

The basic function of the ACT generalization process is to extract out of different special productions what they have in common. These common aspects are embodied in a production that will apply to new situations where original special procedures do not apply. Thus, the claim of the ACT generalization mechanism is that transfer is facilitated if the same components are taught in two procedures so generalization can occur. So, for instance, transfer to a new text editor will be more facilitated if one has studied two other text editors than if one has studied only one.

Another Example. The example above does not illustrate the full complexity at issue in forming generalizations. The following is a fuller illustration of the complexity:

```
P4:        IF the goal is to indicate the relation in (LVobject1 chase LVobject2)
              and LVobject1 is dog
              and LVobject1 is singular
              and LVobject2 is cat
              and LVobject2 is plural
       THEN say CHASES

P5:        IF the goal is to indicate the relation in (LVobject3 scratch LVobject4)
              and LVobject3 is cat
              and LVobject3 is singular
              and LVobject4 is dog
              and LVobject4 is plural
       THEN say SCRATCHES

P6:        IF the goal is to indicate the relation in (LVobject1 LVrelation LVobject2)
              and LVobject1 is singular
              and LVobject2 is plural
```

[2]Throughout this section the language acquisition examples are only meant to illustrate the application of these learning mechanisms. There are many major language acquisition phenomena and issues that are being ignored in this discussion. Anderson (1981) should be consulted for a fuller discussion of how these mechanisms might give a plausible account of some of the phenomena and issues.

```
        THEN say "LVrelation + s"
```

P6 is the generalization that would be formed from P4 and P5. It illustrates that clauses can be deleted in a generalization as well as variables introduced (in this case LVrelation). In this example, the generalization has been made that the verb inflection does not depend on the category of the subject or of the object and does not depend on the verb. This generalization remains overly specific in that the rule still tests that the object is plural--this is something the two examples have in common. Further generalization would be required to delete this unnecessary test. On the other hand, the generalized rule does not test for present tense and so is overly general. This is because this information was not represented in the original productions. The discrimination process, to be described, can bring this missing information in.

The technical work defining generalization in ACT is given in Anderson, Kline, and Beasley (1980) and similar definitions are to be found in Hayes-Roth and McDermott (1976) and Vere (1977). I will skip these technical definitions here for brevity's sake. The basic generalization process is clear without them.

Discipline for Forming Generalizations. In our implementation and in the ACT theory we propose that generalizations are formed whenever two generalizable productions are found in the APPLYLIST. Recall from our earlier discussion (p. xxx) that the APPLYLIST is a probabilistically constituted subset of the system's productions that are potentially relevant to the current situation.

In some situations there are potential generalizations that are technically legal but that seem too risky because they involve too much deletion of constraint from the production condition. For instance, consider the following two productions P7 and P8 and their potential generalization P9:

```
P7:         IF the goal is to indicate LVobject
               and LVobject is a farmer
               and agricol is the word-for farmer
               and LVobject is plural
               and LVobject is in an agentive role
            THEN say "agricol + ae"

P8:         IF the goal is to indicate LVobject
               and LVobject is a girl
               and puell is word-for girl
               and LVobject is singular
               and LVobject possesses another object
            THEN say "puell + ae"

P9:         IF the goal is to indicate LVobject
               and LVobject is a LVclass
               and LVword is the word-for LVclass
            THEN say "LVword + ae"
```

This is a gross overgeneralization but more serious, it violates reasonable constraints on what could possibly be a safe generalization. Basically, the two productions leading to the generalization are too dissimilar. Too much is deleted in going from the specific productions to the generalized production. Currently, we place a limit that no more than 50% of the constants may be lost in the condition by forming a generalization. In the above two productions the constants are underlined. As can be seen only two of the original six constants (6 types, 7 tokens) are preserved in generalization. This is fewer than is acceptable under the 50% rule.

Comparisons to Earlier Conceptions of Generalization. The process of production generalization clearly has similarities to the process of stimulus generalization in earlier learning theories (for a review see Heinemann & Chase, 1975) but there are clear differences also. Past theories frequently proposed that a response conditioned to one stimulus would generalize to stimuli similar on various dimensions. So, for instance, a bar press conditioned to one tone would tend to be evoked by other tones of similar pitch and loudness. An important feature of this earlier conception is that generalization was an automatic outcome of a single learned connection and did not require any further learning. Learning in these theories was all a matter of discrimination--restricting the range of the learned response. In contrast, in the ACT theory generalization is an outcome of comparing two or more learned rules and extracting what they have in common. Thus, it requires additional learning over and above the learning of the initial rules and it depends critically on the relationship between the rules learned. As I will discuss, when we get to the application of these ideas to classification learning, there is evidence for ACT's stronger assumption that generalization depends on the inter-item similarity among the learning experiences as well as the similarity of the test situation to the learning experiences.

Another clear difference between generalization as presented here and many earlier generalization theories is that the current generalization proposed is structural and involves clause deletion and variable creation rather than the creation of ranges on continuous dimensions. We have focused on structural generalizations because of the symbolic domains that have been our concern. However, these generalization mechanisms can be extended to apply to generalization over intervals on continuous dimensions (Brown, 1977; Larson & Michalski, 1977). ACT's generalization ideas are much closer to what happens in stimulus-sampling theory (Burke & Estes, 1957; Estes, 1950) where responses conditioned to one set of stimulus elements can generalize to overlapping sets. This is the same as the notion in ACT of generalization on the basis of clause overlap. However, there is nothing in stimulus-sampling theory that corresponds to ACT's generalization by replacing constants in clauses with variables. This is because stimulus-sampling theory does not have the representational construct of propositions with arguments.

Discrimination

Just as it is necessary to generalize overly specific procedures, so it is necessary to restrict the range of application of overly general procedures. It is possible for productions to become overly general either because of the generalization process or because the critical information was not attended to in the first place. It is for this reason that the discrimination process plays a critical role in the ACT theory. This discrimination process tries to restrict the range of application of productions to just the appropriate circumstances. The discrimination process requires that ACT have examples both of correct and incorrect application of the production. The discrimination algorithm remembers and compares the values of the variables in the correct and incorrect applications. It randomly chooses a variable for discrimination from among those that have different values in the two applications. Having selected a variable, it looks for some attribute which the variable has in only one of the situations. A test is added to the condition of the production for the presence of this attribute.

An Example. An example would serve to illustrate these ideas. Suppose ACT starts out with the following production:

```
P1:          IF the goal is to indicate the relation in (LVsubject LVrelation LVobject)
        THEN say "LVrelation + s"
```

This rule, for generating the present tense singular of a verb is, of course, overly general in the above form. For instance, this rule would apply when the sentence subject was plural, generating "LVrelation + s", when what is wanted is "LVrelation". By comparing circumstances where the above rule applied correctly with the current incorrect situation, ACT could notice that the variable LVsubject was bound to different values and that the value in the correct situation had singular number but the value in the incorrect situation had plural number. ACT can formulate a rule for the current situation that recommends the correct action:

```
P2:          IF the goal is to indicate the relation in (LVsubject LVrelation LVobject)
               and LVsubject is plural
        THEN say "LVrelation"
```

ACT can also form a modification of the previous rule for the past situation:

```
P3:          IF the goal is to indicate the relation in (LVsubject LVrelation LVobject)
               and LVsubject is singular
        THEN say "LVrelation + s"
```

The first discrimination, P2, is called an *action discrimination* because it involves learning a new action while the second discrimination, P3, is called a *condition discrimination* because it involves restricting the condition for the old action. Because of specificity ordering, the action discrimination will block misapplication of the overly general P1. The condition discrimination, P3, is an attempt to reformulate P1 to make it more restrictive. It is important to note that these discriminations do not replace the original production; rather, they coexist with it. ACT can only form an action discrimination when feedback is obtained about the correct action for the situation. If ACT only receives feedback that the old action is incorrect, it only can form a condition discrimination. However, ACT will only form a condition discrimination if the old rule (i.e., P1 in the above example) has achieved a level of strength to indicate that it has some history of success. The reason for this restriction on condition discriminations is that a rule can be formulated that is simply wrong and we do not want to have it perserverate by a process of endlessly proposing new discriminations. Note that productions P2 and P3 are improvements over P1 but are still not sufficiently refined. The *discrimination* algorithm can apply to these, however, comparing where they applied successfully and unsuccessfully. If discriminations of these were formed on the basis of tense and if both response and condition discriminations were formed, we would have the following set of productions:

```
P4:         IF the goal is to indicate the relation in (LVsubject LVrelation LVobject)
               and LVsubject is plural
               and LVrelation has past tense
         THEN say "LVrelation + ed"

P5:         IF the goal is to indicate the relation in (LVsubject LVrelation LVobject)
               and LVsubject is plural
               and LVrelation has present tense
         THEN say "LVrelation"

P6:         IF the goal is to indicate the relation in (LVsubject LVrelation LVobject)
               and LVsubject is singular
               and LVrelation has past tense
         THEN say "LVrelation + ed"

P7:         IF the goal is to indicate the relation in (LVsubject LVrelation LVobject)
               and LVsubject is singular
               and LVrelation has present tense
         THEN say "LVrelation + s"
```

A more thorough consideration of how these mechanisms would apply to acquisition of the verb auxiliary system of English is given in Anderson (1981). The current example is only an illustration of the basic discrimination mechanism.

Recall that the feature selected for discrimination is determined by comparing the variable bindings in the successful and unsuccessful production applications. A variable is selected on which they differ and features are selected to restrict the bindings. It is possible for this discrimination mechanism to choose the wrong variables or wrong features to discriminate on. So, for instance, it may turn out that LVobject has a different number in two circumstances and the system may set out to produce a discrimination on that basis (rather than discriminating on the correct variable, LVsubject). In the case of condition discriminations, such mistakes have no negative impact on the behavior of the system. The discriminated production produces the same behavior as the original in the restricted situation. So it cannot lead to worse behavior. (And recall that the original production still exists to produce the same behavior in other situations.) If an incorrect action discrimination is produced it may block by specificity the correct application of the original production in other situations. However, even here the system can recover by producing the correct discrimination and then giving the correct discrimination a specificity or strength advantage over the incorrect discrimination.

The current discrimination mechanism also attempts to speed up the process of finding useful discriminations by its method of selecting propositions from the data base. Though still using a random process to guarantee that any appropriate propositions in the data will eventually be found, this random choice is biased in certain ways to increase the likelihood of a correct discrimination. The discrimination mechanism chooses propositions with probabilities that vary with their activation levels. The greater the amount of activation that has spread to a proposition, the more likely it is that proposition will be relevant to the current situation.

Feedback and Memory for Past Instances. The previous example illustrated two critical prerequisites for discrimination to work. First the system must receive feedback indicating that a particular production has misapplied and, in the case of an action discrimination, it must receive feedback as to what the correct action should have been. Second, it must remember information about the context of past successful applications of the production. Both assumptions require some further discussion.

In principle, a production application could be characterized as being in one of three states--known to be incorrect, known to be correct, or correctness unknown. However, the mechanisms we have implemented for ACT do not use the distinction between the second and third states. If a production applies and there is no comment on its success, it is treated as if it were a successful application. So the real issue is how ACT identifies that a production application is in error. A production is considered to be in error if it put into working memory a fact that is later tagged as incorrect. There are two basic ways for this error tagging to occur--one is through external feedback and the other is through internal computation. In the external feedback situation the learner may be directly told his behavior is in error or he may infer this by comparing his behavior to an external referent (e.g., the behavior of a model or a textbook answer). In the internal computation case the learner must identify that a fact is contradictory that a goal has failed, or that there is some other failure to meet internal norms. We will discuss later the example of how the learner can use the goal structure in geometry to identify goal-settings that were in error.

The exact details of how feedback is brought to bear on production actions will vary from domain to domain. The issue of negative feedback has been particularly controversial in the domain of language acquisition (e.g., Braine, 1971; Brown, Cazden, & Bellugi, 1969). Given the relative arbitrariness of natural language structures, it is unlikely that internal consistency criteria provide much of a source for detection of production misapplication. The information must come from external sources but it has been argued, with respect to first language acquisition, that negative feedback is rare and not really used when given. However, it is a logical necessity that negative feedback somehow must be brought to bear if the child is to improve his or her generation. Sometimes, this negative feedback may take the form of direct correction of the child's generation. However, in other circumstances it can be more indirect as when a child compares his utterance against that of a present or remembered model utterance. McWhinney (1980) discusses some possibilities for indirect feedback. Whatever the source the child must be capable of identifying a particular piece of utterance as an error and of identifying what the correct utterance should have been (for an action discrimination).

The second issue concerns memory for the context of past utterances. In the actual computer implementations we have stored with each production the context of its last successful application. It seems plausible to suppose that contexts are stored only with certain probabilities, that multiple contexts can be stored, and that contexts are forgotten with increasing delay. This would mean that zero, one, or more contexts might be available to facilitate a discrimination. However, we have not yet developed the empirical base to guide such performance assumptions about memory for past contexts. Rather, we have focused on how a past context should be used if it can be remembered.

Interaction of Discrimination and Specificity. When a discrimination is formed of an overly general rule, the discriminated production does not replace the overly general production; rather, it coexists along with the overly-general production. For many reasons, it is adaptive that the general rule is not thrown out when the discrimination is formed. First, the piece of information that led to the discrimination may have been the wrong one--some other feature was required for discrimination or perhaps no discrimination was required at all. Also, as I will now explain, an overly-general production can participate in a correctly functioning set of productions.

Suppose the system starts out with the following overly general production for generating the 'ed' inflection:

```
P1:        IF the goal is to indicate the relation in (LVsubject LVrelation LVobject)
       THEN say "LVrelation + ed"
```

If this production misapplied in a present, plural context a discrimination would be formed to produce a production adequate to deal with the current context (we will consider for the present example that only action discriminations are formed.) If the discrimination was based on tense the following production would be produced:

```
P2:        IF the goal is to indicate the relation in (LVsubject LVrelation LVobject)
              and LVrelation has present tense
       THEN say "LVrelation"
```

This production would misapply in the singular present context. The following production would be generated by discrimination to produce the right behavior here:

```
P3:        IF the goal is to indicate the relation in (LVsubject LVrelation LVobject)
              and LVrelation has present tense
              and LVsubject is singular
       THEN say "LVrelation + s"
```

Although only this third production is adequately discriminated, a production system consisting of these three productions would generate correct behavior. Production P3 would of course generate the right verb inflection in the present singular context (it would apply rather than P1 or P2 because of specificity). In the context of a present plural, P3 would not match and P2 would take precedence over P1 and correctly generate the null inflection. Finally, in past context P1 would be the only one to apply and correctly generate the 'ed' inflection. Of course, as discussed in Anderson (1981) a richer set of productions would be required to deal with the full verb auxiliary structure of English, but the same interaction between discrimination and specificity can be exploited.

Another case where specificity is exploited has to do with the exceptions to rules. Production P1 would generate the wrong inflection for irregular verbs like *shoot*. A series of discriminations could create a production specific to *shoot* and the past tense--i.e.:

```
P4:         IF the goal is to indicate the relation in (LVsubject shoot LVobject)
               and LVrelation has past tense
         THEN say "SHOT"
```

This production would take precedence over P1 and generate the right form. The presence of this production P4 will not, however, prevent P1 from applying to regular verbs. Note that specific exceptions to general rules, like the production above, will only reliably take precedence over the general rule if they have adequate strength. This may explain why irregular inflectional rules tend to be associated with frequent words.

Strengthening

The generalization and discrimination mechanisms are the inductive components to the learning system in that they are trying to extract from examples of success and failure the features that characterize when a particular production rule is applicable. The generalization and discrimination processes produce multiple variants on the conditions controlling the same action. It is important to realize that at any point in time the system is entertaining as its hypothesis a set of different productions with different conditions to control the action--not just a single production (condition-action rule). There are advantages to be gained in expressive power by means of multiple productions for the same action, differing in condition. Since the features in a production condition are treated conjunctively but separate productions are treated disjunctively, one can express the condition for an action as a disjunction of conjunctions of conditions. Many real world categories have the need for this rather powerful expressive logic. Also, because of specificity ordering, productions can enter into more complex logical relations as we noted.

However, because they are inductive processes sometimes generalization and discrimination will err and produce incorrect productions. There are possibilities for overgeneralizations and useless discriminations. The phenomenon of overgeneralization is well documented in the language acquisition literature occurring both in the learning of syntactic rules and in the learning of natural language concepts. The phenomena of pseudo-discriminations are less well documented in language because a pseudo-discrimination does not lead to incorrect behavior, just unnecessarily restrictive behavior. However, there are some documented cases in the careful analyses of language development (e.g., Maratsos & Chalkley, 1981). One reason that a strength mechanism is needed is because of these inductive failures. It is also the case that the system may simply create productions that are incorrect--either because of misinformation or because of mistakes in its computations. ACT uses its strength mechanism to eliminate wrong productions, whatever their source.

The strength of a production affects the probability that it will be placed on the APPLYLIST and is also used in resolving ties among competing productions of equal specificity on the APPLYLIST. These factors were discussed earlier with respect to the full set of conflict resolution principles in ACT (see p. xxx). ACT has a number of ways of adjusting the strength of a production in order to improve performance. Productions have a strength of .1 when first created. Each time it applies, a production's strength increases by an additive factor of .025. However, when a production applies and receives negative feedback, its strength is reduced by a multiplicative factor of .25. Because a multiplicative adjustment produces a greater change in strength than an additive adjustment, this "punishment" has much more impact than a reinforcement.

Although these two mechanisms are sufficient to adjust the behavior of any fixed set of productions, additional strengthening mechanisms are required to integrate new productions into the behavior of the system. Because these new productions are introduced with low strength, they would seem to be victims of a vicious cycle: They cannot apply unless they are strong, and they are not strong unless they have applied. What is required to break out of this cycle is a means of strengthening productions that does not rely on their actual application. This is achieved by taking all of the strength adjustments that are made to a production that applies and making these adjustments to all of its generalizations as well. Since a general production will be strengthened every time any one of its possibly numerous specializations applies, new generalizations can amass enough strength to extend the range of situations in which ACT performs successfully. Also, because a general production applies more widely, a successful general production will come to gather more strength than its specific variants.

For purposes of strengthening, re-creation of a production that is already in the system, whether by proceduralization, composition, generalization, or discrimination, is treated as equivalent to a successful application. That is, the re-created production receives a .025 strength increment, and so do all of its generalizations.

The exact strengthening values encoded into the ACT system are somewhat arbitrary. The general relationships among the values are certainly important, but the exact relationships are probably not. If all the strength values were multiplied by some scaling factor one would get the same performance from the system. They were selected to give satisfactory performance in a set of language learning examples described by

Anderson, Kline, & Beasley (1980). It is not immediately obvious that they should work to promote the desired productions and suppress the undesirable. However, these parameter settings have worked in a broad range of applications. For instance, we (Anderson, Kline, & Beasley, 1979) were successful in simulating a range of schema abstraction studies (as will be discussed later). I doubt that this is evidence for the exact set of parameters we propose, but this is evidence that these parameter settings are from the right family.

Comparison to Other Discrimination Theories

As in the case of generalization, ACT's mechanisms for discrimination have clear similarities to earlier ideas about discrimination. As in the case with generalization, ACT's discrimination mechanisms focus on structural relations whereas traditional efforts were more focused on continuous dimensions. Brown (1977) has sketched out ways for extending ACT-like discrimination mechanisms to continuous dimensions although we have not developed them in ACT. Also it is the case that ACT discrimination mechanisms are really specified for an operant conditioning paradigm (in that the action of productions are evaluated according to whether they achieve desired behavior and goals) and do not really address the classical conditioning paradigm in which a good deal of research has been done on discrimination. However, despite these major differences in character, a number of interesting connections can be drawn between ACT and the older conceptions of discrimination. In making these comparisons I will be drawing on strengthening and other conflict resolution principles in ACT as well as the discrimination mechanism.

Shift Experiments.

One of the supposedly critical issues in choosing between the discontinuity and continuity theories of discrimination were the shift experiments (Spence, 1940). The paradigm involved taking subjects that were still responding at a chance level with respect to some discrimination (e.g., white-black) and shifting the reinforcement contingencies so that the appropriate response was changed. According to the discontinuity theory the subject's chance performance indicated failure to be entertaining the right hypothesis and the shift should not hurt, while according to the continuity theory the subject could still be building up "habit strength" for the correct response and a shift would hurt. Continuity theory tended to be supported on this issue for infrahuman subjects (e.g., see Kendler & Kendler, 1975). ACT is like the discontinuity theory in that its various productions represent alternative hypotheses about how to solve a problem; however, its predictions are in accord with the continuity theory because it can be accruing strength for a hypothesis before the production is strong enough to apply and produce behavior. Of course, ACT's discrimination mechanisms cannot account for the shift data with adults (e.g., Trabasso & Bower, 1968) but we have argued elsewhere (Anderson, Kline, & Beasley, 1979) that such data should be ascribed to a conscious hypothesis testing process that produces declarative learning rather than an automatic procedural learning process.

Stimulus Generalization and Eventual Discrimination. As noted earlier, the clauses in a production condition are like the elements of stimulus sampling theory. A problem for stimulus sampling theory (see Medin, 1976 for a recent discussion) is how to accomodate both the fact of stimulus generalization and the fact of eventual perfect discrimination. The fact of stimulus generalization can easily be explained in stimulus-sampling theory by assuming that two stimulus conditions overlap in their elements. However, if so, the problem becomes how perfect discrimination behavior can be achieved when the common elements can be associated to the wrong response.

In the ACT theory one can think of the original productions for behavior as basically testing for the null set of elements:

```
P1:        IF the goal is X
         THEN do Y
```

With discrimination, elements can be brought in to discriminate between successful and unsuccessful situations, e.g.:

```
P2:        IF the goal is X
              and B is present
         THEN do Y

P3:        IF the goal is X
              and B is present
              and C is present
         THEN do Z

P4:        IF the goal is X
              and D is present
         THEN do Y

           etc.
```

This is like the conditioning of features to responses in stimulus-sampling theory.

If some features occur sometimes in situations for response Y and sometimes in situations for response Z, discrimination can cause them to become parts of productions recommending one of the actions. For instance, suppose B is such a feature that really does not discriminate between the actions. Suppose B is present in the current situation where response Z is executed but that the system receives feedback indicating that Y is correct. Further, suppose B was not present in the past prior situation where response Z had proved successful. Production P2 would be formed as an action discrimination. The B test is useless because B is just as likely to occur in a Z situation. This corresponds to the *conditioning* of common elements. However, in ACT the strengthening, discrimination, and specificity processes can eventually repress productions that are responding just to common elements. For instance, further discriminative features can be added as in P3 that will serve to block out the incorrect application of P2. Also it is possible to simply weaken P2 and add a new production like P4 which perhaps contains the correct discrimination.

Patterning Effects. The ACT discrimination theory also explains how subjects are trained to give a response in the presence of the stimuli A and B together, but neither A nor B alone. This simply requires that two discriminative clauses be added to the production. Responding to such configural cues was a problem for some of the earlier discrimination theories (see Rudy & Wagner, 1975 for a review). The power of the ACT theory over these early theories is that productions can respond to pattern of elements rather than to each element separately.

ACT also predicts the fact that in the presence of correlated stimuli, one stimulus may partially or completely overshadow a second (see MacKintosh, 1975 for a review). Thus, if both A and B are trained as a correlated pair to response R, one may find that A has less ability to evoke R alone than if it was the only cue associated with R. Sometimes if B is much more salient, A may have no control over R at all. In ACT, the discrimination mechanism will choose among the available features (A, B and other irrelevant stimuli) with probabilities reflecting their salience. Thus, it is possible that a satisfactory discrimination involving B will be found, that this production will be strengthened to where it is dominating behavior and producing satisfactory results, and the A discrimination will never be made. It is also possible that even after a production is formed with the B discrimination, it is too weak to apply, an error occurs, and an A discrimination occurs. In that case both A and B might develop as alternate and equally strong bases for responding. Thus, the ACT theory does not predict that overshadowing will always occur but allows it to occur, and predicts it to be related to the differential salience of the competing stimuli.

Application to Geometry

I assumed that the information in geometry postulates is initially encoded declaratively and intepreted by general productions. In the knowledge compilation section I explained how this declarative representation could be converted into a procedural representation that directly applied the knowledge in the postulate. This produces a rough postulate-to-production correspondence. I will be assuming such a correspondence in my discussion of tuning in geometry. The basic claim will be that production embodiments of postulates become better tuned through practice in their range of application.

The Search Problem. We have developed in some detail how these processes of generalization, discrimination, and strengthening apply in the geometry domain (e.g., Anderson, Greeno, Kline, & Neves, 1981; Anderson, 1981). We feel that it is the tuning provided by these processes which is a major component in the development of expertise in such mathematical-technical domains as geometry. Generating a proof in geometry involves searching a space of possible backward and forward inferences. A striking difference between novices and experts is their better judgment about the right inferences to make.

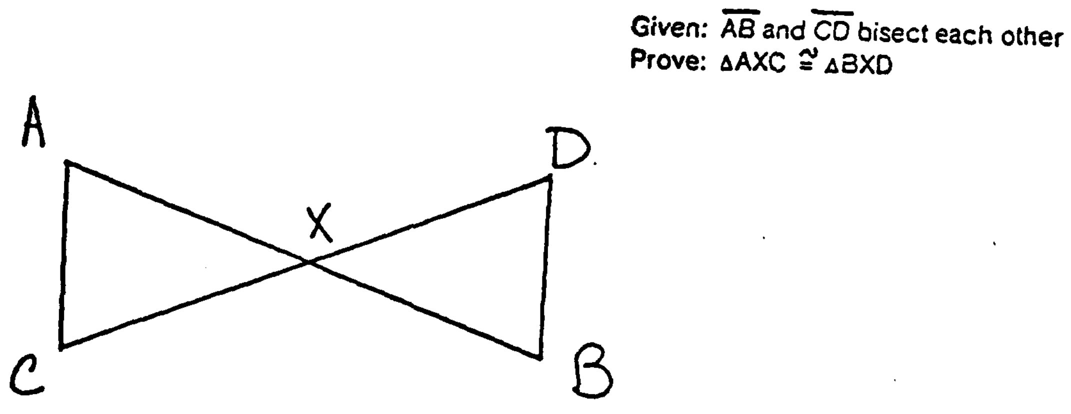
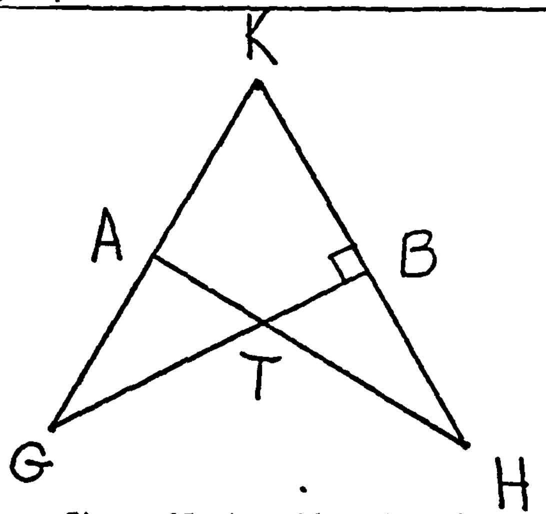

Given: $\overline{AB}$ and $\overline{CD}$ bisect each other
Prove: $\triangle AXC \cong \triangle BXD$
A
D
X
C
B

Figure 14: A problem to which a novice student tried to apply SSS but which experienced students immediately see as involving SAS.

This difference in search judgment can be nicely illustrated by a couple of examples from our geometry protocols. Figure 14 illustrates one of the early triangle-congruence problems that occurs in the text by Jurgensen, Donnelly, Maier, and Rising (1975). One of our students proceeded to try to prove this by means of the SSS (side-side-side) postulate which led him to the subgoal of trying to prove that AC = BD. In contrast, we as instructors have the experience of immediately seeing this as a SAS (side-angle-side) problem. It is not obvious what features we are using to select this method when we see the problem, although it is easy in retrospect to speculate on what features we might have been using. The interesting question, of course, is why the proof method is more available to us than to our student subject.



Given: $\angle GBK$ is a right $\angle$
$\angle H$ is complementary to $\angle K$
$\overline{AK} \cong \overline{BK}$
$\overline{GK} \cong \overline{HK}$
Prove: $\triangle GBK \cong \triangle HAK$

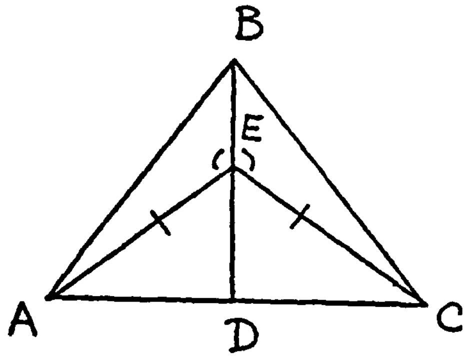
Figure 15: A problem where both novice and experienced students are laid astray as to the optimal proof method.

The problem in Figure 15, which comes from a later section in the chapter on triangle congruence, serves to establish that there is nothing magical about our better judgment in proof direction. It should be noted that this problem came in the section that had introduced the hypotenuse-leg theorem. We and our subject solved it in basically the same way. We used the fact that $\angle H$ is *complementary* to $\angle K$ (i.e., they sum to 90°) and the fact that a triangle has 180° to establish that $\angle HAK$ was 90° and that $\triangle HAK$ was a right angle triangle. Then we could use the two pieces of information given about segment congruence to establish the hypotenuse-leg theorem. However, this problem has a much simpler solution and one that is provided in the teacher's edition of this textbook. Note that one can use the fact that the two triangles share $\angle GKH$ and the two segment congruences to directly permit the SAS postulate. So here is a case where our trained sense about how to proceed in a proof led us astray. This problem violated a number of fairly good heuristics--i.e., always use your givens, use right-angle postulates if right-angle triangles are given, and use the postulates from the current section of the textbook.

A central thesis in our work on geometry is that there are certain features of a problem that are predictive of the success of a particular inference path and that the student learns these correlations between problem features and inference paths through proving problems. Some correlations between problem features and inference rules are logically determined. So, for instance, a student will learn that if he is trying to prove two triangles congruent and they both involve right angles, it is likely that he should try a right angle postulate. Other correlations between problem features and inference rules reflect more about biases in problem construction than any logical necessity. So, for instance, a student learns that if he sees a triangle that looks as if it is isosceles, it is likely that he will want to prove that it is isosceles. Whatever the reason for the correlation between features and inference methods, the student can use these feature-method correlations as heuristics to guide search. An important task for our tuning mechanisms is to discover and exploit these correlations. It is clear that students become more judicious in choice of proof paths because they learn more and more features that are predictive of the correct paths.

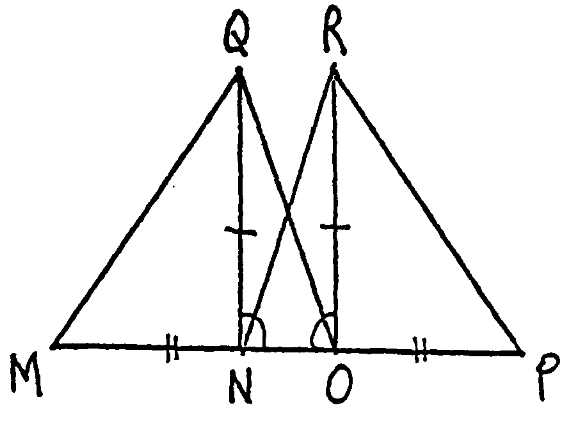
Generalization. We have worked out some simulations of the application of generalization and discrimination to form improved rules. Figure 16 illustrates a fairly powerful example of generalization at work. Although these two problems seem quite different they do allow a generalization. From working on individual problems like these, subjects can compile productions that recommend the successful method for solving the problem. So, for these two problems, subjects might create:

(a)



Given: $\overline{EA} \cong \overline{CE}$
$\angle BEA \cong \angle BEC$
Prove: $\triangle ABD \cong \triangle CBD$

(b)



Given: $\overline{QN} \cong \overline{OR}$
$\angle ONQ \cong \angle NOR$
$\overline{MN} \cong \overline{OP}$
Prove: $\triangle QOM \cong \triangle RNP$

Figure 16: By generalizing specific operators for (a) and (b) the student can form a more powerful operator.

P1: IF the goal is to prove $\triangle ABD \cong \triangle CBD$
and they contain $\triangle ABE$ and $\triangle CBE$
and $\overline{EA} \cong \overline{EC}$
and $\angle BEA \cong \angle BEC$
THEN set as a subgoal to prove $\triangle ABE \cong \triangle CBE$

P2: IF the goal is to prove $\triangle QOM \cong \triangle RNP$
and they contain $\triangle QON$ and $\triangle RNO$
and $\overline{NQ} \cong \overline{OR}$
and $\angle ONQ \cong \angle NOR$
THEN set as a subgoal to prove $\triangle QON \cong \triangle RNO$

It should be clear that these two productions are distinct and not just notational variants of one another. The first describes two triangles that share a side and there are lines meeting at a common point (E) on that side to define two contained triangles. The two triangles in P2 only partially overlap on one side and two other triangles share that overlap. Despite these differences these two productions can be generalized to create the following production:

P3: IF the goal is to prove $\triangle XYZ \cong \triangle UVW$
and they contain $\triangle XYS$ and $\triangle UVT$
and $\overline{SX} \cong \overline{TU}$
and $\angle YSX \cong \angle VTR$
THEN set as a subgoal to prove $\triangle XYS \cong \triangle UVT$

where we have the following variable equivalences:

P3	*P1*	*P2*
X	A	Q
Y	B	O
Z	D	M
U	C	R
V	B	N
W	D	P
S	E	N
T	E	O

This generalized production embodies the rule that if the goal is to prove a pair of triangles congruent, and they share sides with a second pair of triangles, and the second pair has a congruent side and angle, then set as a subgoal to prove the second pair congruent. The generalization has the same basic character of the language acquisition examples given earlier, viz., it creates a general production whose condition preserves what the original productions have in common.

Discrimination. Figure 17 illustrates an example of discrimination where we can compare our subject's performance with the simulation. In Part (a) we have one problem our subject solved and in Part (b) we have the next problem. The first problem was solved by SSS and this experience apparently primed SSS in our subject because in the next problem he tried the method of SSS which fails and only then did he try SAS which succeeded. In ACT the failure of SSS is the stimulus for the discrimination process. The production that had applied in this case might be a general encoding of the SSS postulate which we can represent:

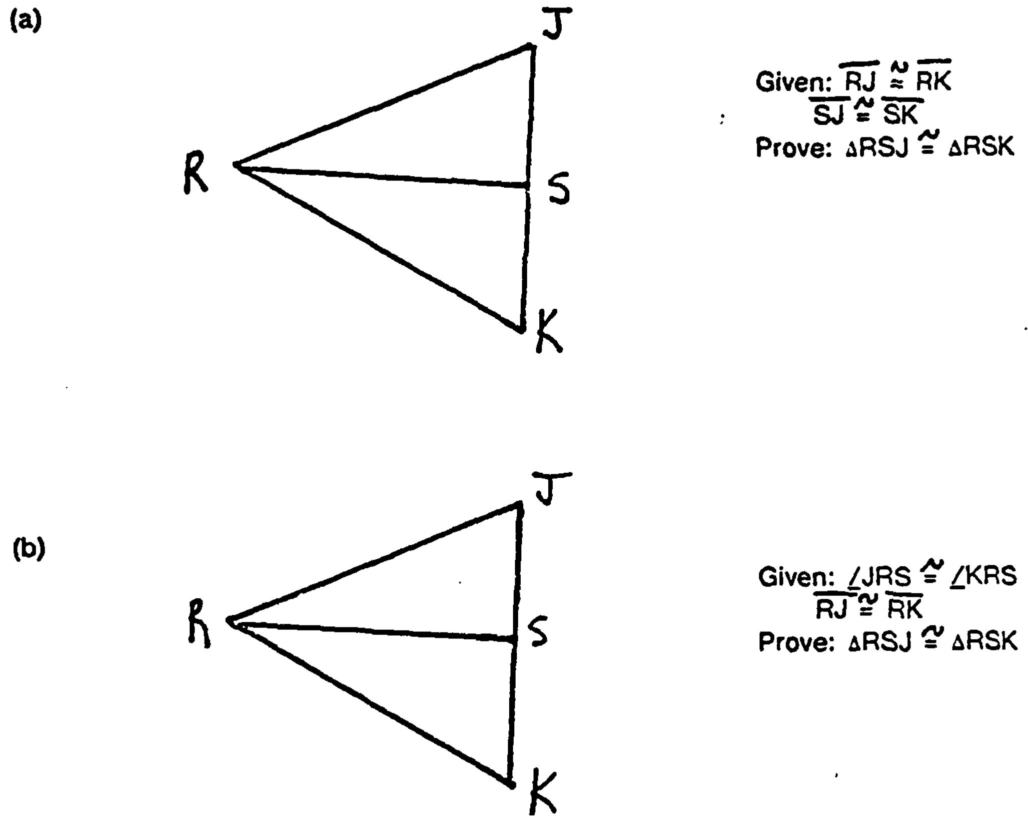

(a)
J
R
S
K
Given: $\overline{RJ} \cong \overline{RK}$
$\overline{SJ} \cong \overline{SK}$
Prove: $\Delta RSJ \cong \Delta RSK$
(b)
J
R
S
K
Given: $\angle JRS \cong \angle KRS$
$\overline{RJ} \cong \overline{RK}$
Prove: $\Delta RSJ \cong \Delta RSK$

Figure 17: By comparing (a) where SSS works with (b) where it does not and SAS does, the student can create more discriminate productions for the application of SSS and SAS.

P1: IF the goal is to prove $\Delta XYZ \cong \Delta UVW$
THEN try to prove this by means of SSS

The system would compare the failure on this problem (b) to an earlier success on the earlier similar problem in part (a). As in our earlier discussion of discrimination there are two discriminations that ACT can create. It can form a condition discrimination to restrict SSS to the type of situation in Figure 17a. For instance, that problem mentioned two side congruences whereas Figure 17b does not. This would lead to the following production.

P2: IF the goal is to prove $\Delta XYZ \cong \Delta UVW$
and $\overline{XY} \cong \overline{UV}$
and $\overline{YZ} \cong \overline{VW}$
THEN try to prove this by means of SSS

or ACT can form an action discrimination that will recommend SSS for the current situation. One distinctive feature is that an angle congruence is mentioned:

P3: IF the goal is to prove $\Delta XYZ \cong \Delta UVW$
and $\angle XYZ \cong \angle UVW$
THEN try to prove this by means of SAS

Both of these discriminations appear to be steps in the direction of more adequate heuristics. In fact, our subject remarked after this example that he thought he should not try SSS as a proof method when angles were mentioned. This is evidence for the comparison process assumed by the discrimination mechanism.

By continued generalizations and discriminations and by adjusting their strengths according to their success, the system can develop a very rich characterization of the problem types that appear and of the appropriate response to each problem type. Basically, we propose that what happens in geometry is like the pattern learning that is purported to occur in the acquisition of chess skill (Chase & Simon, 1973; Simon & Gilmartin, 1975) where it is claimed that chess masters have acquired on the order of 50,000 critical patterns and have associated an appropriate line of response with each pattern. I would like to suggest that the tuning process discussed here for geometry underlay the acquisition of these chess rules. The patterns are formed from direct encodings of chess positions and from discriminations and generalizations derived from these.

Credit-Blame Assignment in Geometry. There is an interesting issue of credit-blame assignment in any interesting problem-solving situation--be it geometry or chess. After ACT has completed a proof it has a goal structure reflecting the process that led to the proof. It can identify which goals in that goal structure were successful and which were failures. Productions that led to the creation of failed portions of the search net are regarded as having misapplied in that they led the system away from its goal. These are the ones that are subjects for discrimination. A little care is required to properly identify the erroneous productions. As an example, suppose a goal is set to prove two angles congruent by showing that they are corresponding parts of congruent triangles. Suppose all methods tried for proving congruent triangles fail and angle congruence is eventually proven by resorting to the supplementary angle postulate. The mistake is not in the methods attempted for proving triangles congruent; rather, the mistake was in setting the subgoal of triangle congruence. ACT's credit-blame assignment procedure would correctly identify the point of error. This is an example where the hierarchical goal structure of behavior is used critically to aid the learning process.

Composition. The composition idea that we developed as part of our model of knowledge compilation can also be used to package sequences of inference steps into single macro-operators. A somewhat similar idea in the domain of logic proofs has been advanced by Smith (in press). Figure 18 illustrates one of the problems where we applied this mechanism. The first pass of this system over the problem was accomplished by a sequence of three productions.

P1: IF the <u>goal</u> is <u>to prove</u> $\underline{\Delta XYZ} \cong \Delta UVW$
and $\overline{XY} \cong \overline{UV}$ and $\overline{YZ} \cong \overline{VW}$
THEN set as a subgoal to prove $\angle XYZ \cong \angle UVW$

P2: IF the <u>goal is</u> to prove $\angle XYZ \cong \angle UYW$
and $\overline{XYW}$ and $\overline{UYZ}$
THEN this can be concluded by vertical angles

P3: IF the goal is to prove $\Delta XYZ \cong \Delta UVW$

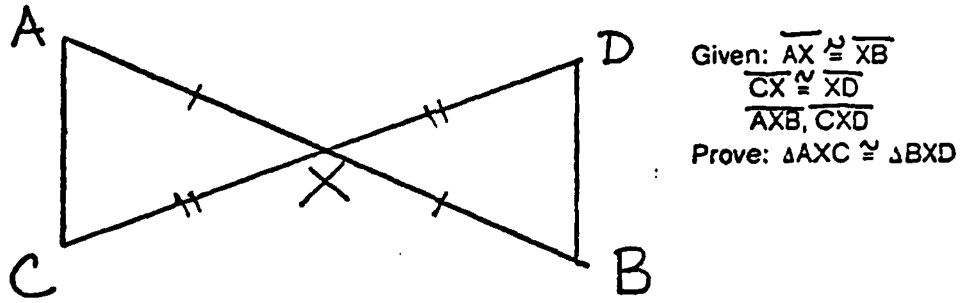

A
D
X
C
B
Given: $\overline{AX} \cong \overline{XB}$
$\overline{CX} \cong \overline{XD}$
$\overline{AXB}, \overline{CXD}$
Prove: $\triangle AXC \cong \triangle BXD$

Figure 18: The pattern represented in this example occurs with such frequency that some students have compiled a production to recognize it.

and $\overline{XY} \cong \overline{UV}$, $\overline{YZ} \cong \overline{VW}$, and $\angle XYZ \cong \angle UVW$
THEN this can be concluded by SAS

Production P1 recognizes that there are two pairs of congruent sides and sets the goal to prove the included angles congruent. Production P2 recognizes the vertical angles pattern and that the two angles are therefore congruent. Production P3 recognizes that all the components are now available for the SAS postulate to apply. Composing these three productions together we get:

P4: IF the goal is to prove $\triangle XYZ \cong \triangle UYW$
and $\overline{XY} \cong \overline{UV}$ and $\overline{YZ} \cong \overline{YW}$
and $\overline{XYW}$ and $\overline{UYZ}$
THEN conclude $\angle XYZ \cong \angle UYW$ by vertical angles
and conclude $\triangle XYZ \cong \triangle UYW$ by SAS

Creation of Data-Driven Productions. It is a feature of the composed production P4 that it summarizes what had been a multi-level goal tree. The system had started with the goal of proving two triangles congruent, set a subgoal of proving two angles congruent, and then proceeded to pop the goal. Production P4 will only apply if the goal is explicitly set to prove the two triangles congruent. However, the situation described in the condition of P4 is so special that even if the goal had not been explicitly set, it would be useful to make the inference to embellish the problem. Certainly subjects can be observed to make such "forward inferences" independent of current goals. ACT can create a forward-inference or data-driven production by dropping the goal specification from P4 (a similar idea was proposed by Larkin, 1981). The resulting production would be:

P5: IF there are $\triangle XYZ$ and $\triangle UYW$
$\overline{XY} \cong \overline{UY}$ and $\overline{YZ} \cong \overline{YW}$
and $\overline{XYW}$ and $\overline{UYZ}$
THEN conclude $\angle XYZ \cong \angle UYW$ by vertical angles
and $\triangle XYZ \cong \triangle UYW$ by SAS

Forward inferences can be made when composition creates a macro-operator which achieves a stated goal by a sequence of inferences that previously had involved the embedding of subgoals. The forward inference can be created from the composition by deleting the goal clause. It is useful to understand why one would only want to drop goal clauses from the macro-operators rather than the original working-backwards productions. The original productions are so little constrained that the goal clauses provide important additional tests of applicability. After a macro-operator is composed there are enough tests in the non-goal aspects of its condition to make it quite likely that the inferences will be useful. That is, it is unlikely to be an accident that the conjunction of tests are satisfied. There is clear evidence for such a forward inference rule like P5 in the protocols of some of the more advanced students. For them, the pattern in Figure 18 is something that will trigger the set of inferences even when it appears embedded in a larger problem.

Procedural Learning: The Power Law

One aspect of skill acquisition is distinguished both by its ubiquity and its surface contradiction to ACT's multiple-stage, multiple-mechanism view of skill development. This is the log-linear or power law for practice: A plot of the logarithm of time to perform a task against the logarithm of amount of practice is a straight line, more or less. It has been widely discussed with respect to human performance (Fitts & Posner, 1967; Welford, 1968) and has been the subject of a number of recent theoretical analyses (Lewis, 1979; Newell & Rosenbloom, 1981). It is found in phenomena as diverse as motor skills (Snoddy, 1926), pattern recognition (Neiser, Novich, & Losar, 1963), problem-solving (Neves & Anderson, 1981), memory retrieval (Anderson, in preparation), and suspiciously, in machine-building by industrial plants (an example of institutional learning not human learning--Hirsch, 1952). Figure 19 illustrates one example--the effect of practice on the speed with which inverted text can be read (Kolers, 1975). This ubiquitous phenomenon would seem to contradict the ACT theory of skill acquisition because at first it seems that a theory which proposes changing mechanisms of skill acquisition would not predict the apparent uniformity of the speed up. Also it is not clear immediately why ACT would predict a power function rather than, say, an exponential function. Because of the ubiquity of the power law, it is important to show the ACT learning theory is consistent with this phenomenon.

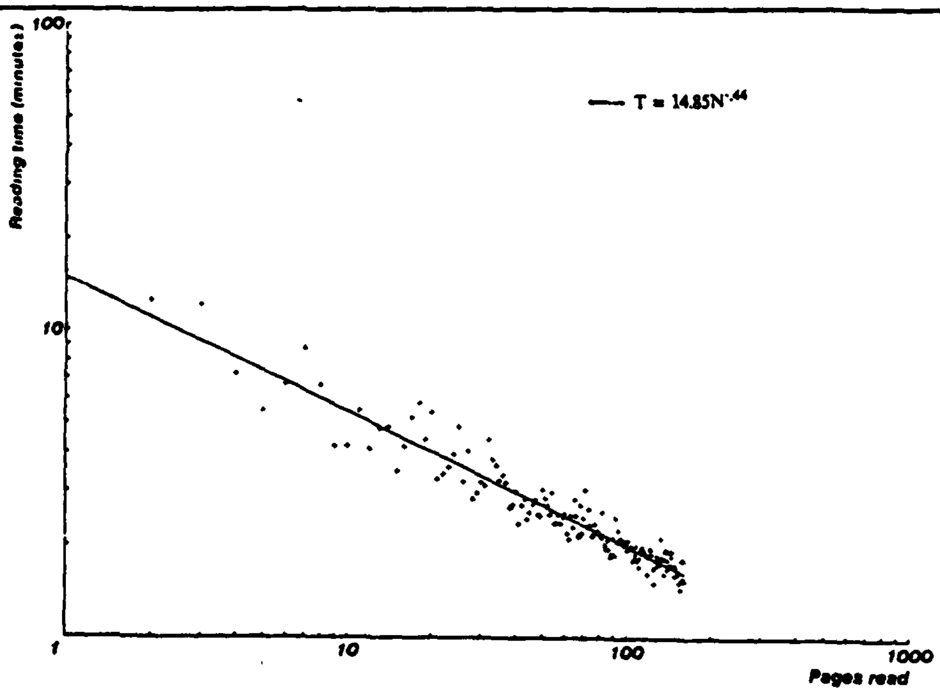


Figure 19: The effect of practice on the speed with which subjects can read inverted text--from Kolers, 1975.

The general form of the equation relating time (T) to perform a task to amount of practice (P) is

$$T = X + AP^{-b} \tag{1}$$

where X is the asymptotic speed, X + A is the speed on trial 1 and b is the slope of the function on a log-log plot (where time is plotted as ln(T-X)). The asymptotic X is usually very small relative to X + A and the rate of approach to asymptote is slow in a power function. This means that it is possible to get very good fits in plots like Figure 19 assuming a zero asymptote. However, careful analysis of data with enough practice does indicate evidence for non-zero asymptotes.

These facts about skill speed-up have appeared contradictory to ACT-like learning mechanisms because ACT mechanisms would seem to imply speed-up faster than a power law. For instance, it was noted (p. xxx) that composition seemed to predict a speed-up on the order of Ba^P--which is to say an exponential function of practice, P (*a* is less than 1). An exponential law, as noted by Newell and Rosenbloom (1981), is in some sense the natural prediction about speed-up. It assumes that with each practice trial the subject can improve a constant fraction (*a*) of his current time or that he has a constant probability each trial of a constant fraction of improvement. When we look at ACT's tuning mechanisms of discrimination and generalization, it is harder to make general claims about the speed-up they will produce because their speed-up will depend on the characteristics of the problem space. However, it is at least plausible to propose that each discrimination or generalization has a constant expected factor of improvement. Composition, generalization, and discrimination improve performance by reducing the expected number of productions applied in performing a task. I will refer to improvement due to reduction in number of productions as *algorithmic improvement.*

In contrast to algorithmic improvement, strengthening reduces the time for individual productions of the procedure to apply. I will show that the strengthening process in ACT does result in a power law. However, even if strengthening obeys a power law it is not immediately obvious why the total processing, which is a product of both algorithmic improvement and strengthening, should obey a power law. Nonetheless, I will set forth a set of assumptions under which this is just what is predicted by ACT and in so doing will resolve the problem.

Strengthening

While complex processes like editing or proof generation appear to obey a power law it is also the case that simple processes like simple choice-reaction time (Mowbray & Rhoades, 1954) or memory retrieval (Anderson, in preparation) appear to obey a power law. In these cases the speed-up cannot be modelled as an algorithmic improvement in number of production steps. There cannot be more than a small number of productions (e.g., 10) applying in the less than 500 msec required for these tasks. A process reducing that number would not produce the continuous improvements observed. Moreover, subjects may well start out with optimal or near optimal procedures in terms of minimum number of productions. So there often is little or no room for algorithmic improvement. Here we have to assume that the speed-up observed is due to a basic increase in the rate of production application as would be produced by ACT's strengthening process.

Recall from our earlier discussions (p. xxx) that time to apply a production is c + a/s where *s* is the production strength, *c* reflects processes in production application, and *a* is the time for a unit-strength production to be selected. Strength increases one unit (a unit is arbitrarily .025 in our theory) with each trial of practice. Therefore, we can simply replace *s* in the above by *P*, the number of trials of practice. Then, the form of the practice function for production execution in ACT would seem to be:

$$T = c + aP^{-1} \tag{2}$$

which is a hyperbolic function, one form of the power law. This assumes that on the first measured trial (P=1), the production already has 1 unit of strength from an earlier encoding opportunity. The time for N such productions to apply would be:

$$T = cN + aNP^{-1} \tag{3}$$

$$\text{or } T = C + AP^{-1} \tag{4}$$

This is a power law where the exponent is 1 and the asymptote C. The problem is that, unless peculiar assumptions are made about prior practice (see Newell & Rosenbloom, 1981), the exponent obtained is typically much less than 1 (usually in the range .1 to .6).

However, the smaller exponents are to be predicted when one takes into account that there is forgetting or loss of strength from prior practice. Thus, a better form of Equation (4) would be:

$$T = C + A/\sum_{i=0}^{P-1} s(i,P) \tag{5}$$

where s(i,P) denotes the strength remaining from the ith strengthening when the Pth trial comes about. In the above S(0,P) denotes the strength on trial P of the initial encoding trial. To understand the behavior of this function we have to understand the behavior of the critical sum

$$S = \sum_{i=0}^{P-1} s\,(i,P) \tag{6}$$

Wickelgren (1976) has shown that the strength of the memory trace decays as a power law. Assuming that time is linear in number of practice trials we have:

$$s(i,P) = D\,(P-i)^{-d} \tag{7}$$

where D is the initial strength and $d < 1$. Combining (6) and (7) we get:

$$S = \sum_{i=1}^{P} Di^{-d} \tag{8}$$

This function is bounded below and above as follows:

$$\frac{D}{1-d}\left((P+1)^{1-d} - 1\right) < S < \frac{D}{1-d}\left(P^{1-d} - d\right) \tag{9}$$

S is closely approximated by the average of these upper and lower bounds and since the difference between $(P+1)^{1-d}$ and P^{1-d} becomes increasingly small with large *P* we may write

$$S \approx \frac{D}{1-d}\left(P^{1-d} - X\right)$$

where $X = (1+d)/2$. So, the important observation is that, to a close approximation, total strength will grow as a power law. Substituting back into equation (5) we get

$$T = C' + A'P^{-g} \tag{10}$$

where $A' = A(1-d)/D$, $g = 1-d$, and $C' = C - DX/(1-d)$. Thus, the ACT model predicts that time for a fixed sequence of productions should decrease as a power law with the exponent deviating from 1 (and a hyperbolic function) to the degree that there is forgetting. The basic prediction of a power function is confirmed in simple tasks; the further prediction relating the exponent to forgetting is a difficult issue requiring further research. However, it is known that forgetting does reduce the effect of practice (e.g., Kolers, 1975). Given that forgetting must be an important factor in the long-term development of a skill, the ACT analysis of the power law is at a distinct advantage over other analyses which do not accomodate forgetting effects.

Algorithmic Improvement

There is an interesting relationship between this power law for simple tasks, based just on strength accumulation, and the power law for complex tasks where there is also the potential for reduction in number of production steps. We noted in the case of composition that a limit on this process was that all the information to be matched by the composed production must be active in working memory. Because the size of production conditions (despite the optimization produced by proceduralization) tends to increase exponentially with compositions, the requirements on working memory for the next composition tend to increase exponentially with the number of compositions. It is also the case that, as successful discriminations and generalizations proceed, there will be an increase in the amount of information that needs to be held in working memory so that another useful feature can be identified. In this case, it is not possible to make precise statements concerning the factor of increase but it is not unreasonable to suppose that this increase is also exponential with number of improvements. This then implies that the following relationship should define the size (W) of working memory needed for the ith algorithmic improvement.

$$W = GH^i \qquad (11)$$

where G and H are the parameters of the exponential function.

The ACT theory predicts that there should be a power law describing the amount of activation of a knowledge structure as a function of practice (in the concepts or links that define that structure). By the same analysis as the one just given for production strength, ACT predicts that the strength of memory structures should increase as a power function of practice. The strength of a memory structure directly determines the amount of activation it will receive. Thus, we have the following equation describing total memory activation (A) as a function of practice:

$$A = QP^r \qquad (12)$$

where Q and r are the parameters of the power function. (Note that *P* is raised to a positive exponent, *r*, less than one.) This equation is more than just theoretical speculation; unpublished work in our laboratory on effects of practice on memory retrieval has confirmed this relationship.

There is a strong relationship in the ACT theory between the working memory requirements described by Equation (11) and the total activation described by Equation (12). For an amount *W* of information to be available in working memory the information must reach a threshold level of activation *L* which means that the total amount of activation of the information structure will be described by:

$$A = WL \qquad (13)$$

Equations (11), (12), and (13) may be combined to derive a relationship between the number of improvements (i) and amount of practice:

$$i = r\frac{\ln(P)}{\ln(H)} + \frac{\ln(Q)}{\ln(H)} - \frac{\ln(L)}{\ln(H)} - \frac{\ln(G)}{\ln(H)} \qquad (14)$$

or more simply

$$i = r\frac{\ln(P)}{\ln(H)} + X \qquad (15)$$

Thus, because of working memory limitations, the rate of algorithmic improvement is a logarithmic rather than a linear function of practice. Continuing with the assumption that the number of steps (N) should be reduced by a constant fraction f with each improvement we get:

$$N = N_0 f^i \tag{16}$$

or

$$N = N_0' P^{-f'} \tag{17}$$

where

$$f' = -\frac{r \ln(f)}{\ln(H)} \quad \text{and} \quad N_0' = N_0 f^x \tag{18}$$

Thus, the number of productions to be applied should decrease as a power function of practice. Equation (17) assumes that in the limit, 0 steps are required to perform the task but there must be some minimum N* which is the optimal procedure. Exactly, how to introduce this minimum into Equation (16) will depend on one's analysis of the improvements, but if we simply add it, we will get the standard power function for improvement to an asymptote:

$$N = N^* + N_0' P^{-f'} \tag{19}$$

So let us review the analysis of the power law to date. We started with the observation that, assuming that the rate of algorithmic improvement is linear with practice and that each improvement has a proportional decrease in number of productions, an exponential practice function is predicted, not a power practice function. We noted that the mechanisms of strength accumulation predict that individual productions should speed up as a power function. Similar strength dynamics governing the growth of working memory size imply that the rate of algorithmic improvement was actually logarithmic and therefore the decrease in number of productions would be a power function.

It should be noted that the relationship between working memory capacity and improvements in the production algorithm corresponds to a common subject experience on complex tasks. Initially, subjects report feeling swamped trying to just keep up with the task and have no sense of the overall organization of the task. With practice subjects report beginning to perceive the structure of the task and claim to be able to see how to make improvements. It is certainly the case that we observe subjects better able to maintain current state and goal and better able to retrieve past goals and states of the task. Thus, it seems that their working memory for the problem improves with practice and subjects claim that being able to apprehend at once a substantial portion of the problem is what is critical to making improvements.

Algorithmic Improvement and Strengthening Combined

The total time to perform a task is determined by the number of productions and the time per production. Therefore, the simplest prediction about total time (TT) would be to combine multiplicatively Equation (10) describing time per production and Equation (19) describing number of productions:

$$TT = [N^* + N_0' P^{-f'}][C' + A' P^{-g}] \tag{20}$$

Because of the asymptotic components, N* and A', the above will not be a pure power law but it will look like a power function to a good approximation (as good an approximation as is typically observed empirically). If N* and C were 0, then we would have a pure power law of the form:

$$TT = N_0' A' P^{-(f'+g)} \tag{21}$$

This has a zero asymptote. Because the initial time is so large relative to final time, most data are fit very well assuming a 0 asymptote. This is the form of the equation we will use for further discussion.

One complication ignored in the foregoing discussion is that algorithmic improvements in the number of productions typically mean creation of new productions. According to the theory, new productions start off with low strength. Thus, productions at later points in the experiment will not have been practiced since the beginning of the experiment and will have lower strength than assumed in equations (20) and (21). Another complication on top of this is that a completely new set of productions will not be instituted with each improvement, only a subset will change. Suppose that at any point in time the productions in use were introduced an average of *j* improvements ago. This means (by equation 15) that after the ith improvement the average production has been practiced from trial KL^{i-j} to trial KL^i and therefore has had $KL^i(1-L^{-j})$ trials of practice where $K = H^{-X/r}$ and $L = H^{1/r}$ from equation (15). Thus, the number of trials of practice (P*) for a production is expected to be a constant fraction of the total number of trials (P) on the task:

$$P^* = qP \tag{22}$$

where $q = (1 - L^j)$. This implies that the correct form of equation 20 is

$$TT = N_0' A' q^{-g} P^{-(f'+g)} \tag{23}$$

Thus, this does not at all affect the expectation of a power function.

An Experimental Test

The basic prediction of this analysis is that both number of productions and time per production should decrease as a power function of practice. As a result total time will decrease as a power function. Neves and Anderson (1981) have tested this prediction in an experiment that studied subjects' ability to give reasons for the lines of an abstract logic proof. This reason-giving task is modelled after a frequent kind of exercise found in high school geometry texts (see Fig. 3). However, we wanted to use the task with college students and wanted to see the effects of practice starting from the beginning. Therefore, we invented a novel artificial proof system. Each proof consisted of 10 lines. Each line could be justified as a given or derived from earlier lines by application of one of nine postulates. Subjects only could see the current line of the proof and had to request of a computer that particular prior lines, givens, or postulates be displayed. The method of requesting this information was very easy and so we hoped to be able to trace, by subjects' request behavior, the steps of the algorithm that they were following. The relationship between requests and production application is almost certainly one-to-many, but we believe that we can use these requests as an index of the number of productions that are applying. The basic assumption is that the ratio of productions to requests will not change over time. This assumption certainly could be challenged but I think it is not implausible and is strongly supported by the orderliness of the results. Under this assumption, if we plot number of requests as a function of practice we are looking at the reduction in the number of productions or algorithmic improvement. If we plot time per request we are looking at the improvement in the speed of individual productions.

Figure 20 presents the analysis of this data averaged over three subjects (individual subjects show the same pattern). Subjects took about 25 minutes to do the first problem. After 90 problems they were often taking under 2 minutes to do the proofs. This reflects the impact of approximately 10 hours of practice. As can be seen from Figure 20 both number of steps (information requests) and time per step (interval between requests) go down as power functions of practice. Hence, total time also obeys a power function. The exponents for the number of steps is -.346 (varies from -.315 to -.373 for individual subjects) while the exponent for the time per step is -.198 (range -.144 to -.226).

The Neves and Anderson experiment does provide evidence that underlying a power law in complex tasks are power laws both in number of steps applied and in time per step. I have shown how a power law in strength accumulation may underlie both of these phenomena. While it is true that algorithmic improvement would tend to produce exponential speed-up, the underlying strength dynamics determine working memory capacity and produce a power function in algorithmic improvement. It is natural to think of these strength dynamics as describing a process at the neural level of the system. Therefore, it is interesting to note Eccles' (1972) review of the evidence that individual neurons increase with practice in their rate of transmitter release and pickup across synapses and that they decrease with disuse.

Tracing the Course of Skill Learning: The Classification Task

We have given separate analyses to the declarative and procedural stages of skill performance and we have given separate analyses to the learning mechanisms that produce the transition between stages and to the learning mechanisms applying in the procedural stage. To indicate the combined effect of these many mechanisms, I would like to consider the development of a rather simple skill from beginning to end. This is the ability to classify objects as belonging to a particular category. There is a fairly active experimental literature (e.g., Brooks, 1978; Franks & Bransford, 1971; Hayes-Roth & Hayes-Roth, 1977; Medin & Schaeffer, 1978; Newman, 1974; Posner & Keele, 1970; Reed, 1972; Reitman & Bower, 1973; Rosch &

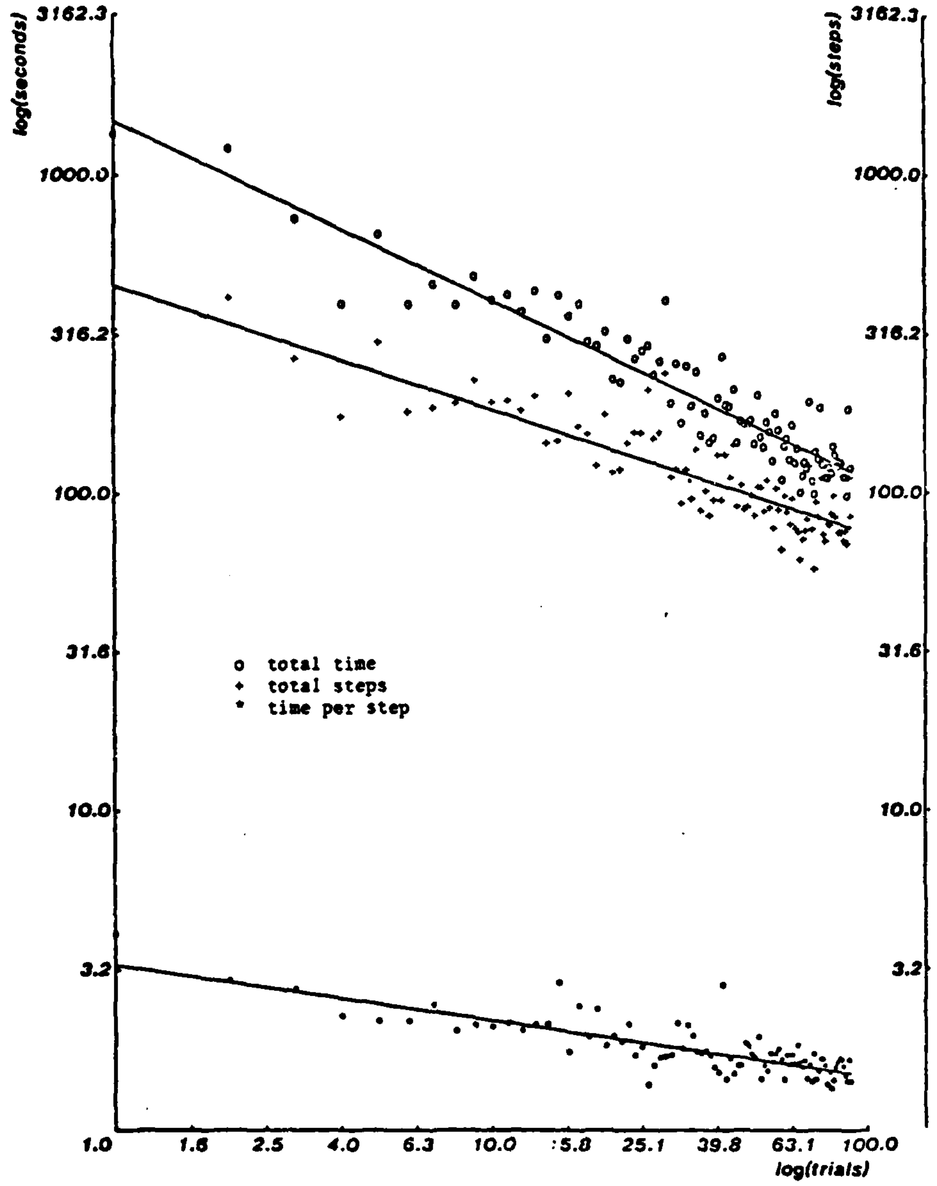


Figure 20: The effect of practice on a reason giving task. Plotted separately are the effects on number of steps, time per step, and total time--from Neves and Anderson, 1981.

Mervis, 1975) concerned with this phenomena which is typically called prototype formation or schema abstraction. The experimental task is very simple: subjects are presented with stimuli that vary on a number of dimensions and they must learn to categorize them into a number of categories. The categories tend to be formed according to complex rules or tend only to statistically approximate a rule. Subjects' efforts to identify the categories by deliberate rule induction tend not to be very successful (Brooks, 1978; Reber, 1967); however, subjects do manage to extract some of the regularities from the set. Subjects often report that they make their classification on some general sense of similarity to other stimuli. The typical experiment involves a training stage in which subjects are trained to classify some set of exemplars until they reach a fairly high level of performance. They then go to a transfer task in which they are asked to classify new instances.

Evidence for the fact that they have extracted regularities from the initial set come from the reliable manner in which they can assign new instances to categories.

Initial Performance

Subjects can do this task after instruction as simple as:

> "You will see a sequence of descriptions of individuals. Your task is to learn to assign them to category 1 or category 2."

Since such instruction does no more than specify the goal, it seems clear that subjects must call on an already existing subroutine to perform the task. The hypothesis of a prior subroutine is certainly plausible in that this is not the first time that students would be asked to assign instances to categories.

Table 5 provides a model of what the initial procedure might be (I have identified, for later use, the variables in these productions.) The procedure in Table 5 assigns a new instance to a category by trying to retrieve a known instance that is similar to the new instance. P1 starts the processing by selecting an instance for consideration. Since highly similar instances will overlap in features, a spreading activation mechanism for selecting past instances would tend to select a similar instance. That is, presentation of a test instance would activate its features and highly similar instances in memory would be selected at the intersection of activation from these features. However as we will see, even if a similar instance is not selected first it can be selected later. The production P1 sets the goal of comparing the similarity of the presented and remembered item. It sets to 0 a counter which will provide a measure of similarity. Production P5 increments the counter when one of the presented item's features is shared by the remembered item; P6 leaves the counter unchanged if no value can be remembered on one of the dimensions of the presented item; P7 decrements the counter when a contradiction in features is found; P8 notes when all the features of the current stimulus have been checked and returns control to the higher routine. If the counter exceeds some criterion C, production P2 will classify the new item as being in the same category as the old item; if not P3 will select a new item for testing; if there are no more items that can be recalled for testing P4 will randomly choose a category to assign the item to.

The production set in Table 5 can be thought of as implementing a procedure of classifying new examples by analogy to past examples. This scheme for pattern classification is very much like that of Medin and Schaffer (1978). Medin and Schaffer showed how such an instance-based categorization system can account for many of the results in the schema abstraction literature. I consider the production system in Table 4 to be an adequate model for performance in early stages of the classification task.

Application of Knowledge Compilation

It is useful to consider how proceduralization and composition would apply to the production set in Table 5 for classification. Suppose a subject is in a setting where he must classify person descriptions as members of Club 1 or Club 2 and he is presented with the following instance--married, Catholic, plays tennis, and has gone to trade school. Suppose (via Production P1 in Table 5) the subject recalls a Club 1 member who is married, Catholic, bowls, and went to trade school. If all the attributes of item 3 were remembered the sequence of productions to apply from Table 3 would be P1 to create the subgoal of feature comparison; then P2 would apply three times matching marital status, religion, and education; and production P4 would apply to note that tennis and bowling do not match. Following this P5 would pop the comparison goal and P6

Table 5
An Initial Set of Productions for Performing Classifications

```
P1:      IF the goal is to classify LVitem1
            and LVitem2 is a past instance
       THEN set as a subgoal to compare LVitem1 and LVitem2
            and set the similarity counter to 0

P2:      IF the goal is to classify LVitem1
            and LVitem1 has been compared to LVitem2
            and the similarity counter has value greater than C
            and LVitem2 belonged to LVcategory
       THEN assign LVitem1 to LVcategory
            and POP the goal

P3:      IF the goal is to classify LVitem1
            and LVitem1 has been compared to LVitem2
            and the counter is less than C
            and LVitem3 is a past instance
       THEN set as a subgoal to compare LVitem1 to LVitem3
            and set the counter to 0

P4:      IF the goal is to classify LVitem1
            and there are no more past instances
            and LVcategory is a category of the experiment
       THEN assign LVitem1 to LVcategory
            and POP the goal

P5:      IF the goal is to compare LVitem1 and LVitem2
            and LVitem1 has LVvalue on LVdimension
            and LVitem2 has LVvalue on LVdimension
       THEN increment the similarity counter

P6:      IF the goal is to compare LVitem1 and LVitem2
            and LVitem1 has LVvalue on LVdimension
            and there is no value remembered for LVitem2 on LVdimension
       THEN continue

P7:      IF the goal is to compare LVitem1 and LVitem2
            and LVitem1 has LVvalue1 on LVdimension
            and LVitem2 has LVvalue2 in LVdimension
            and LVvalue1 ≠ LVvalue2
       THEN decrement the similarity counter

P8:      IF the goal is to compare LVitem1 and LVitem2
            and there are no more dimensions to compare for LVitem1
       THEN POP the goal
```

would assign the item to Club 1 on the basis of the overlapping features. If this sequence were repeated often enough and each time resulted in a successful classification, the eventual product of composition and proceduralization would be:

```
        IF the goal is to classify LViteml
           and LViteml is married
           and LViteml is Catholic
           and LViteml has gone to trade school
           and LViteml bowls
      THEN assign LViteml to Club 1
```

(In this composition we have dropped the use of the match counter as something only needed by the subroutine.) Thus, the impact of composition and proceduralization is to create productions that basically contain complete descriptions of the instances or nearly complete descriptions (if only some features are remembered). Replacing the productions in Table 5 by productions like the above may not change the behavior of the system in terms of its classification choices. However, it certainly speeds up the classification process. Putting the instance information into production form is also critical in that it puts the knowledge into a form in which the tuning processes (to be described next) of generalization, discrimination, and strengthening can apply.

Tuning of the Classification Productions

Our efforts to model classification behavior are particularly relevant to evaluating the learning mechanisms of generalization, discrimination, and strengthening. Anderson, Kline, and Beasley (1979) present an account of the application of these ACT learning mechanisms to the schema abstraction domain. Here I will summarize that application and indicate the essential features about the ACT tuning mechanisms that are responsible for the success of the theory in accounting for the data. As just discussed, the output of the knowledge compilation process will be a set of productions which produce categorization of the specific stimuli on which training has occurred. So, for instance, we might have a pair of productions such as

```
P1:     IF the goal is to classify LViteml
           and LViteml is married
           and LViteml is Catholic
           and LViteml has gone to trade school
           and LViteml plays tennis
      THEN assign LViteml to Club 1

P2:     IF the goal is to classify LViteml
           and LViteml is married
           and LViteml is Catholic
           and LViteml has gone to trade school
           and LViteml plays golf
      THEN assign LViteml to Club 1
```

Generalizing these two productions we get

```
P3:     IF the goal is to classify LViteml
```

```
          and LViteml is married
          and LViteml is Catholic
          and LViteml had gone to trade school
     THEN assign LViteml to Club 1
```

This is a production which "predicts" that this is a Club 1 item on the basis of three features.

The above illustrates the application of generalization to this domain: now consider the application of discrimination. Suppose by generalization or partial compilation we had the following production:

```
P4:        IF the goal is to classify LViteml
              and LViteml is Baptist
              and LViteml plays chess
           THEN assign the item to Club 1
```

Suppose that this rule correctly assigns a married Baptist with college education who plays chess to Club 1 but incorrectly assigns to Club 1 a married Baptist with high school education who plays chess (which is identified as a Club 2 item). This would lead to the following pair of discriminations:

```
P5:        IF the goal is to classify LViteml
              and LViteml is Baptist
              and LViteml plays chess
              and LViteml has college education
           THEN assign LViteml to Club 1

P6:        IF the goal is to classify LViteml
              and LViteml is Baptist
              and LViteml plays chess
              and LViteml is high school educated
           THEN assign LViteml to Club 2
```

In Anderson, Kline, and Beasley we let these generalizations and discriminations occur in response to experience with the examples. A strength mechanism served to weight the various currently competing rules. We used these basic three mechanisms (generalization, discrimination, and strengthening) to simulate the schema abstraction results of Franks and Bransford (1971), Hayes-Roth and Hayes-Roth (1977), and Medin and Schaffer (1978). The simulations were quite good--they fit the data at least as well as did the theories proposed in the original papers that reported the data. (ACT's success at predicting these results depended heavily on its generalization and strengthening mechanism but not on its discrimination processes.)

I would like to note here three essential aspects of the data and how ACT accounted for each aspect. First, there is often a tendency for subjects to most accurately and confidently classify instances closest to the overall central tendency of the category. Sometimes subjects will classify non-studied central instances more accurately than studied, non-central instances. This is predicted by ACT. Since there tend to be more similar instances around the central tendency, ACT will usually form more generalizations that can classify central instances. However, there are a couple of important exceptions to this central tendency effect. Sometimes subjects will perform better on non-central frequent items than on less frequent central items. This is because frequent presentations of non-central items increase the strength of productions that will classify them. Finally, subjects sometimes rate non-central items more highly than central items if there are some study items highly similar to the non-central items but no study items highly similar to the central item. (An item can be at the center of a category but not particularly close to any studied item.) This result is predicted by ACT because generalizations will be formed from the similar study instances to classify the non-central item but this will not happen for the central item. I think it is to ACT's credit that it accomodates the balance of the central tendency effect with the other factors.

ACT is one of the feature-set theories of schema abstraction (others include Hayes-Roth & Hayes-Roth, 1977; Reitman & Bower, 1973). These models assume that subjects learn information about sets of features that occur in the stimulus set. ACT is to be distinguished from these other models because of the special role it gives to generalization as a basis for identifying feature sets. Recently, Elio and Anderson (in press) performed a series of experiments to see what evidence there might be for this special role for generalizations. We had some subjects study pairs of instances of a category such as

(1) Baptist, plays golf, works for government, college-educated, and single
(2) Baptist, plays golf, works for private firm, college-educated, and married

which support a generalization (Baptist, plays golf, and college- educated). Other subjects studied pairs of instances like

(3) Baptist, plays golf, unemployed, high-school educated, and single
(4) Baptist, plays tennis, works for private firm, college-educated and divorced

which do not support much of a generalization. Regardless of which group they were in subjects were tested on transfer pairs such as:

(5) Baptist, plays golf, unemployed, college-educated, divorced

It is important to note that (5) overlaps with (1) and (2) on three features each and it similarly overlaps with (3) and (4) on three features each. Thus, according to other feature set theories there should be identical transfer from both stimulus sets. However, according to the ACT theory there will be better transfer from the condition where subjects are trained on (1) and (2). In fact, the ACT predictions were confirmed.

Summary

We have now reviewed the basic progression of skill acquisition according to the ACT learning theory--it starts out as the interpretive application of declarative knowledge; this becomes compiled into a procedural form, and this procedural form undergoes a process of continual refinement of conditions and raw increase in speed. In a sense this is a stage analysis of human learning. Much as other stage analyses of human behavior, this stage analysis of ACT is being offered as an approximation to characterize a rather complex system of interactions. Any interesting behavior is produced by a set of elementary components and different components can be at different stages. For instance, part of a task can be performed interpretively while another part is performed compiled.

The claim is that this configuration of learning mechanisms described is involved in the full range of skill acquisition from language acquisition to problem solving to schema abstraction. Another strong claim is that the basic control architecture across these situations is hierarchical, goal-structured, and basically organized for problem solving. This echoes the claim made elsewhere (Newell, 1980) that problem solving is the basic mode of cognition. The claim is that the mechanisms of skill acquisition basically function within the mold provided by the basic problem-solving character of skills. As skills evolve they become more tuned and compiled and the original search of the problem space may drop out as a significant aspect. I presented a variety of theoretical analyses and special experimental tests that provide positive evidence for this broad view of skill acquisition. Clearly, many more analyses and experimental tests can be done. However, I think the available evidence at least conveys a modest degree of credibility to the theory presented.

In conclusion, I would like to point out that the learning theory proposed here has achieved a unique accomplishment. Unlike past learning theories it has cogently addressed the issue of how symbolic or cognitive skills are acquired. (Indeed, I have been so focused on this, I have ignored some of the phenomena that traditional learning addressed such as classical conditioning.) The inadequacies of past learning theories to account for symbolic behavior have been a major source of criticism. On the other hand, unlike many of the current cognitive theories, ACT not only provides an analysis of the performance of a cognitive skill but also an analysis of its acquisition. Many researchers (e.g., Estes, 1975; Langley & Simon, 1981; Rumelhart & Norman, 1978) have lamented how the strides in task analysis within cognitive psychology have not been accompanied by strides in development of learning theory.

If I were to select the conceptual developments most essential to this theory of the acquisition of cognitive skills, I would point to two. First, there is the clear separation made in ACT between declarative knowledge (propositional network of facts) and procedural knowledge (production system). The declarative system has the capacity to represent abstract facts. The production system through its use of variables can process the propositional character of this data base. Also, productions through their reference to goal structures have the capacity to shift attention and control in a symbolic way. These basic symbolic capacities are what are essential to the success of the learning mechanisms. Knowledge is integrated into the system by first being encoded declaratively and being interpreted. We argued that the successful integration of knowledge into behavior requires that it first go through such an interpretive stage. The various learning mechanisms all are structured around variable use and reference to goal structures. Moreover, the learning processes impact on the course of the symbolic processing making it both faster and more judicious in choice. In ACT we see how learning and symbolic processing could be synergetic. These two aspects of cognition surely are synergetic in man and this fact commends the theory for consideration at least as much as any specific issue that we considered.

The second essential development is the ACT production system architecture itself. Productions are relatively simple and well-defined objects and this is essential if one is to produce general learning mechanisms. The general learning mechanisms must be constituted so that they will correctly operate on the full range of structures (productions) that they might encounter. It is possible to construct such learning mechanisms for ACT productions; it would not be possible if the procedural formalism were something as diverse and unconstrained as LISP functions. ACT productions have the virtue of S-R bonds with respect to their simplicity, but also have considerable computational power. A problem with many production system formalisms with respect to learning is that it is hard for the learning mechanism to appreciate the function of the production in the overall flow of control. This is why the use of goal structures is such a significant augmentation to the ACT architecture. By inspecting the goal structure in which a production application participates, it is possible to understand the role of the production. This is essential to a system that learns by doing.

References

Anderson, J.R. *Language, Memory, and Thought.* Hillsdale, N.J.: Lawrence Erlbaum Associates, 1976.

Anderson, J.R. *Cognitive Psychology and its Implications.* San Francisco, CA: W.H. Freeman and Company, 1980.

Anderson, J.R. A theory of language acquisition based on general learning mechanisms. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, 1981.

Anderson, J.R. Tuning of search of the problem space for geometry proofs. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, 1981.

Anderson, J.R. Effects of Practice on Memory Retrieval, in preparation.

Anderson, J.R., Greeno, J.G., Kline, P.J., & Neves, D.M. Acquisition of problem-solving skill. In J.R. Anderson (Ed.), *Cognitive Skills and their Acquisition*, Hillsdale, N.J.: Lawrence Erlbaum Associates, 1981.

Anderson, J.R., Kline, P.J., & Beasley, C.M. A theory of the acquisition of cognitive skills. ONR Technical Report 77-1, Yale University, 1977.

Anderson, J.R., Kline, P.J., & Beasley, C.M. A general learning theory and its application to schema abstraction. In G.H. Bower (Ed.), *The Psychology of Learning and Motivation, Vol. 13*, New York, NY: Academic Press, 1979, 277-318.

Anderson, J.R., Kline, P.J., & Beasley, C.M. Complex learning processes. In R.E. Snow, P.A. Federico, & W.E. Montague (Eds.), *Aptitude, Learning, and Instruction; Vol. 2*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1980.

Book, W.F. The psychology of skill with special reference to its acquisition in typewriting. Missoula, Montana: University of Montana, 1908. Facsimile in *The Psychology of Skill*, New York: Armo Press, 1973.

Braine, M.D.S. On learning grammatical order of words. *Psychological Review*, 1963, 70, 323-348.

Braine, M.D.S. On two types of models of the internalization of grammars. In D.I. Slobin (Ed.), *The Ontogenesis of Grammar*. New York: Academic Press, 1971.

Briggs, G.E. & Blaha, J. Memory retrieval and central comparison times in information-processing. *Journal of Experimental Psychology*, 1969, 79, 395-402.

Brooks, L. Nonanalytic concept formation and memory for instances. In E. Rosch & B.B. Lloyd (Eds.), *Cognition and Categorization*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1978.

Brown, D.J.H. Concept learning by feature value interval abstraction. In the Proceedings of the Workshop on Pattern-Directed Inference Systems, 1977.

Brown, J.S. & Van Lehn, K. Repair theory: A generative theory of bugs in procedural skills. *Cognitive Science*, 1980, 4, 379-426.

Brown, R. *A first language*. Cambridge, Mass.: Harvard University Press, 1973.

Brown, R., Cazden, C.G., & Bellugi, V. The child's grammar from I to III. In R. Brown (Ed.), *Psycholinguistics*. New York: The Free Press, 1970, 100-154.

Burke, C.J. & Estes, W.K. A component model for stimulus variables in discrimination learning. *Psychometrika*, 1957, 22, 133-145.

Chase, W.G. & Simon, H.A. The mind's eye in chess. In W.G. Chase (Ed.), *Visual Information Processing*, New York, NY: Academic Press, 1973.

Eccles, J.C. Possible synaptic mechanisms subserving learning. In A.G. Karyman and J.C. Eccles (Eds.), *Brain and Human Behavior*, New York: Springer-Verlag, 1972.

Elio, R. & Anderson, J.R. Effects of category generalizations and instance similarity on schema abstraction. *Journal of Experimental Psychology: Human Learning and Memory*. In press.

Estes, W.K. Toward a statistical theory of learning. *Psychological Review*, 1950, 57, 94-107.

Estes, W.K. The state of the field: General problems and issues of theory and metatheory. In W.K. Estes (Ed.), *Handbook of Learning and Cognitive Processes, Vol. 1*, 1975.

Fitts, P.M. Perceptual-motor skill learning. In A.W. Melton (Ed.), *Categories of human learning*, New York: Academic Press.

Fitts, P.M. & Posner, M.I. *Human Performance*. Belmont, CA: Brooks Cole, 1967.

Forgy, C. & McDermott, J. OPS, a domain-independent production system.

Proceedings of the Fifth International Joint Conference on Artificial Intelligence. 1977, 933-939.

Franks, J.J. & Bransford, J.D. Abstraction of visual patterns. *Journal of Experimental Psychology*, 1971, **90**, 65-74.

Hayes-Roth, B. & Hayes-Roth, F. Concept learning and the recognition and classification of exemplars. *Journal of Verbal Learning and Verbal Behavior*, 1977, **16**, 321-338.

Hayes-Roth, F. & McDermott, J. Learning structured patterns from examples. *Proceedings of the Third International Joint Conference on Pattern Recognition*, 1976, 419-423.

Heinemann, E.C. & Chase, S. Stimulus generalization. In W.K. Estes (Ed.), *Handbook of Learning and Cognitive Processes, Vol. 2*, Hillsdale, N.J.: Lawrence Erlbaum Associates, 1975.

Hirsch, W.Z. Manufacturing progress functions. *Review of Economics and Statistics*, 1952, **34**, 143-155.

Jurgensen, R.C., Donnelly, A.J., Maier, J.E., & Rising, G.R. *Geometry*. Boston, MA: Houghton Mifflin, 1975.

Kendler, H.H. & Kendler, T.S. From discrimination learning to cognitive development: A neobehavioristic odyssey. In W.K. Estes (Ed.), *Handbook of Learning and Cognitive Processes, Vol. 1*, Hillsdale, N.J.: Lawrence Erlbaum Associates, 1975.

Kline, P.J. The superiority of relative criteria in partial matching and generalization. *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, 1981.

Kolers, P.A. Reading a year later. *Journal of Experimental Psychology: Human Learning and Memory*, 1975, **1**, 689-701.

Langley, P. & Simon, H.A. The central role of learning in cognition. In J.R. Anderson (Ed.), *Cognitive Skills and their Acquisition*. Hillsdale, N.J.: Lawrence Erlbaum Associates, 1981.

Larkin, J.H. Enriching formal knowledge: A model for learning to solve textbook physics problems. In J.R. Anderson (Ed.), *Cognitive Skills and their Acquisition*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1981.

Larkin, J.H., McDermott, J., Simon, D.P., & Simon, H.A. Expert and novice performance in solving physics problems. *Science*, 1980, **208**, 1335-1342.

Larson, J. & Michalski, R.S. Inductive inference of VL decision rules. In the Proceedings of the Workship on Pattern-Directed Inference Systems, 1977.

Lewis, C.H. Production system models of practice effects. Unpublished dissertation. University of Michigan, Ann Arbor, MI, 1978.

Lewis, C.H. Speed and practice. Unpublished manuscript, 1979.

Luchins, A.S. Mechanization in problem solving. *Psychological Monographs*, 1942, 54, No. 248.

Luchins, A.S. & Luchins, E.H. *Rigidity of behavior; a variational approach to the effect of Einstellung.* Eugene, OR: University of Oregon Books, 1959.

MacKintosh, N.J. From classical conditioning to discrimination learning. In W.K. Estes (Ed.), *Handbook of Learning and Cognitive Processes, Vol. 1*, Hillsdale, N.J.: Lawrence Erlbaum Associates, 1975.

MacWhinney, B. Basic syntactic processes. In S. Kuczaj (Ed.), *Language development: Syntax and Semantics.* Hillsdale, N.J.: Lawrence Erlbaum Associates, 1980.

Maratsos, M.P. & Chalkley, M.A. The internal language of children's syntax: The ontogenesis and representation of syntactic categories. In K. Nelson (Ed.), *Children's Language Vol. 1.* New York, NY: Gardner Press, 1981.

McNeill, D. On theories of language acquisition. In T.R. Dixon & D.L. Horton (Eds.), *Verbal Behavior and General Behavior Theory.* Englewood Cliffs, NJ: Prentice-Hall, 1968.

Medin, D.L. Theories of discrimination learning and learning set. In W.K. Estes (Ed.), *Handbook of Learning and Cognitive Processes, Vol. 3.* Hillsdale, N.J.: Lawrence Erlbaum Associates, 1976.

Medin, D.L. & Schaffer, M.M. A context theory of classification learning. *Psychological Review*, 1978, 85, 207-238.

Miller, G.A., Galanter, E., & Pribram, K.H. *Plans and the Structure of Behavior.* New York, NY: Holt, Rinehard, and Winston, Inc., 1960.

Mowbray, G.H. & Rhoades, M.V. On the reduction of choice reaction times with practice. *Quarterly Journal of Experimental Psychology*, 1959, 11, 16-23.

Neisser, U., Novick, R., & Lazar, R. Searching for ten targets simultaneously. *Perceptual and Math Skills*, 1963, 17, 955-961.

Neumann, P.G. An attribute frequency model for the abstraction of prototypes. *Memory and Cognition*, 1974, 2, 241-248.

Neves, D.M. Learning procedures from examples. Unpublished doctoral dissertation. Carnegie-Mellon University, 1981.

Neves, D.M. & Anderson, J.R. Knowledge compilation: Mechanisms for the automatization of cognitive skills. In J.R. Anderson (Ed.), *Cognitive Skills and their Acquisition*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1981.

Newell, A. Reasoning, problem-solving, and decision processes: The problem space as a fundamental category. In R. Nickerson (Ed.), *Attention and Performance VIII*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1980.

Newell, A. & Rosenbloom, P. Mechanisms of skill acquisition and the law of practice. In J.R. Anderson (Ed.), *Cognitive Skills and their Acquisition*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1981.

Norman, D.A. Discussion: Teaching, learning, and the representation of knowledge. In R.E. Snow, P.A. Federico, and W.E. Montague (Eds.), *Aptitude, Learning, and Instruction, Vol. 2*, Hillsdale, N.J.: Lawrence Erlbaum Associates, 1980.

Posner, M.I. & Keele, S.W. Retention of abstract ideas. *Journal of Experimental Psychology*, 1970, 83, 304-308.

Reber, A.S. Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior*, 1967, 6, 855-863.

Reed, S., Pattern recognition and categorization. *Cognitive Psychology*, 1972, 3, 382-407.

Reitman, J.S. & Bower, G.H. Structure and later recognition of exemplars of concepts. *Cognitive Psychology*, 1973, 4, 194-206.

Rosch, E. & Mervis, C.B. Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, 1975, 7, 573-605.

Rudy, J.W. & Wagner, A.R. Stimulus selection on associative learning. In W.K. Estes (Ed.), *Handbook of Learning and Cognitive Processes, Vol. 2*, Hillsdale, N.J.: Lawrence Erlbaum Associates, 1975.

Rumelhart, D.E. & Norman, D.A. Accretion, tuning, and restructuring: Three modes of learning. In J.W. Cotton & R. Klatzsky (Eds.), *Semantic factors in cognition.* Hillsdale, NJ: Lawrence Erlbaum Associates, 1978.

Rychener, M.D. Approaches to knowledge acquisition: The instructable production system project, 1981.

Rychener, M.D. & Newell, A. An instructible production system: Basic design issues. In D.A. Waterman & F. Hayes-Roth (Eds.), *Pattern-Directed Inference Systems*, New York, NY: Academic Press, 1978.

Schneider, W. & Shiffrin, R.M. Controlled and automatic human information processing: I. Detection, search, and attention. *Psychological Review*, 1977, **84**, 1-66.

Shiffrin, R.M. & Dumais, S.T. The development of automatism. In J.R. Anderson (Ed.), *Cognitive Skills and their Acquisition*, Hillsdale, N.J.: Lawrence Erlbaum Associates, 1981.

Shiffrin, R.M. & Schneider, W. Controlled and automatic human information processing: II. Perceptual learning, automatic attending, and a general theory. *Psychological Review*, 1977, **84**, 127-190.

Simon, H.A. & Gilmartin, K. A simulation of memory for chess positions. *Cognitive Psychology*, 1973, **5**, 29-46.

Sternberg, S. Memory scanning: Mental processes revealed by reaction time experiments. *American Scientist*, 1969, **57**, 421-457.

Trabasso, T.R. & Bower, G.H. *Attention in Learning.* New York, NY: John Wiley, 1968.

Vere, S.A. Inductive learning of relational productions. *Proceedings of the Workshop on Pattern-Directed Inference*, Hawaii, 1977.

Welford, A.T. *Fundamentals of skill.* London: Melhuen, 1968.

Wickelgren, W.A. Memory storage dynamics. In W.K. Estes (Ed.), *Handbook of Learning and Cognitive Processes, Vol. 4.* Hillsdale, NJ: Lawrence Erlbaum Associates, 1976.

DISTRIBUTION LIST

Navy

Dr. Ed Aiken
Navy Personnel R&D Center
San Diego, CA 92152

Meryl S. Baker
NPRDC
Code P309
San Diego, CA 92152

Dr. Robert Breaux
Code N-711
NAVTRAEQUIPCEN
Orlando, FL 32813

Dr. Richard Elster
Dept. of Administrative Sciences
Naval Postgraduate School
Monterey, CA 93940

Dr. Pat Federico
Navy Personnel R&D Center
San Diego, CA 92152

Dr. John Ford
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Henry M. Halff
Department of Psychology, C-009
University of California at San Diego
La Jolla, CA 92093

Lt. Steven D. Harris, MSC, USN
Code 6021
Naval Air Development Center
Warminster, PA 18974

Dr. Jim Hollan
Code 304
Navy Personnel R&D Center
San Diego, CA 92152

Cdr. Charles W. Hutchins
Naval Air Systems Command Hq
AIR-340F
Navy Department
Washington, D.C. 20361

Cdr. Robert S. Kennedy
Head, Human Performance Sciences
Navel Aerospace Medical Research Lab
Box 29407
New Orleans, LA 70189

Dr. Norman J. Kerr
Chief of Naval Technical Training
Naval Air Station Memphis (75)
Millington, TN 38054

Dr. William L. Maloy
Principal Civilian Advisor for
Education and Training
Naval Training Command, Code 00A
Pensacola, FL 32508

Dr. Kneale Marshall
Scientific Advisor to DCNO(MPT)
OP01T
Washington, D.C. 20370

Capt. Richard L. Martin, USN
Prospective Command Officer
USS Carl Vinson (CVN-70)
Newport News Shipbuilding and Drydock Co.
Newport News, VA 23607

Dr. James McBride
Navy Personnel R&D Center
San Diego, CA 92152

Dr. William Montague
Navy Personnel R&D Center
San Diego, CA 92152

Ted M.I. Yellen
Technical Information Office, Code 201
Navy Personnel R&D Center
San Diego, CA 92152

Library, Code P 201L
Navy Personnel R&D Center
San Diego, CA 92152

Technical Director
Navy Personnel R&D Center
San Diego, CA 92152

Commanding Officer
Naval Research Laboratory
Code 2627
Washington, D.C. 20390

Psychologist
ONR Branch Office
Bldg. 114, Section D
666 Summer Street
Boston, MA 02210

Psychologist
ONR Branch Office
536 S. Clark Street
Chicago, IL 60605

Office of Naval Research
Code 437
800 N. Quincy Street
Arlington, VA 22217

Psychologist
ONR Branch Office
1030 East Green Street
Pasadena, CA 91101

Special Asst. for Education and
Training (OP-01E)
Rm. 2705 Arlington Annex
Washington, D.C. 20370

Office of the Chief of Naval Operations
Research Development & Studies Branch
(OP-115)
Washington, D.C. 20350

Dr. Donald F. Parker
Graduate School of Business Administration
University of Michigan
Ann Arbor, MI 48109

Personnel & Training Research Programs
(Code 458)
Office of Naval Research
Arlington, VA 22217

Lt. Frank C. Petho, MSC, USN (Ph.D.)
Selection and Training Research Div.
Human Performance Sciences Dept.
Naval Aerospace Medical Research Lab
Pensacola, FL 32508

Dr. Gary Poock
Operations Research Department
Code 55PK
Naval Postgraduate School
Monterey, CA 93940

Roger W. Remington, Ph.D.
Code L52
NAMRL
Pensacola, FL 32508

Dr. Bernard Rimland (03B)
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Worth Scanland, Director
Research, Development, Test & Evaluation
N-5
Naval Education and Training Command
NAS, Pensacola, FL 32508

Dr. Robert G. Smith
Office of Chief of Naval Operations
OP-987H
Washington, D.C. 20350

Dr. Alfred F. Smode
Training Analysis & Evaluation Group
(TAEG)
Department of the Navy
Orlando, FL 32813

Dr. Richard Sorensen
Navy Personnel R&D Center
San Diego, CA 92152

Roger Weissinger-Baylon
Department of Administrative Sciences
Naval Postgraduate School
Monterey, CA 93940

AD-A103 283   CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF PSYCHOLOGY   F/G 5/10
ACQUISITION OF COGNITIVE SKILL.(U)
AUG 81 J R ANDERSON   N00014-81-C-0335
UNCLASSIFIED TR-81-1   NL

2 of 2
AD A
103283

END
DATE
FILMED
10-81
DTIC

Dr. Robert Wisher
Code 309
Navy Personnel R&D Center
San Diego, CA 92152

Mr. John H. Wolfe
Code P310
U.S. Navy Personnel Research and
Development Center
San Diego, CA 92152

Army

Technical Director
U.S. Army Research Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Beatrice J. Farr
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Dexter Fletcher
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Frank J. Harris
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Col. Frank Hart
Army Research Institute for the
Behavioral and Social Sciences
5001 Eisenhower Blvd.
Alexandria, VA 22333

Dr. Michael Kaplan
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Milton S. Katz
Training Technical Area
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Harold F. O'Neil, Jr.
Attn: PERI-OK
Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Robert Sasmor
U.S. Army Research Institute for the
Behavioral and Social Sciences
5001 Eisenhower Avenue
Alexandria, VA 22333

Dr. Frederick Steinheiser
Department of the Navy
Chief of Naval Operations
OP-113
Washington, D.C. 20350

Dr. Joseph Ward
U.S. Army Research Institute
5001 Eisenhower Avenue
Alexandria, VA 22333

Air Force

Dr. Earl A. Alluisi
HQ, AFHRL (AFSC)
Brooks AFB, TX 78235

Dr. Genevieve Haddad
Program Manager
Life Sciences Directorate
AFOSR
Bolling AFB, D.C. 20332

Dr. Marty Rockway
Technical Director
AFHRL(OT)
Williams AFB, AZ 58224

3700 TCHTW/TTGH Stop 32
Sheppard AFB, TX 76311

Marines

H. William Greenup
Education Advisor (E031)
Education Center, MCDEC
Quantico, VA 22134

Special Assistant for Marine
Corps Matters
Code 100M
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217

Dr. A.L. Slafkosky
Scientific Advisor (Code RD-1)
HQ, U.S. Marine Corps
Washington, D.C. 20380

Other DoD

Defense Technical Information Center
Cameron Station, Bldg. 5
Alexandria, VA 22314
Attn: TC

Dr. Craig I. Fields
Advanced Research Projects Agency
1400 Wilson Blvd.
Arlington, VA 22209

Military Assistant for Training and
Personnel Technology
Office of the Under Secretary of Defense
for Research and Engineering
Room 3D129, The Pentagon
Washington, D.C. 20301

DARPA
1400 Wilson Blvd.
Arlington, VA 22209

Civil Govt

Dr. Susan Chipman
Learning and Development
National Institute of Education
1200 19th Street NW
Washington, D.C. 20208

William J. McLaurin
66610 Howie Court
Camp Springs, MD 20031

Dr. Arthur Melmed
National Institute of Education
1200 19th Street NW
Washington, D.C. 20208

Dr. Andrew R. Molnar
Science Education Development
and Research
National Science Foundation
Washington, D.C. 20550

Dr. Joseph Psotka
National Institute of Education
1200 19th Street NW
Washington, D.C. 20208

Dr. Frank Withrow
U.S. Office of Education
400 Maryland Avenue SW
Washington, D.C. 20202

Dr. Joseph L. Young, Director
Memory and Cognitive Processes
National Science Foundation
Washington, D.C. 20550

Non Govt

Anderson, Thomas H., Ph.D.
Center for the Study of Reading
174 Children's Research Center
51 Gerty Drive
Champaign, IL 61820

Dr. John Annett
Department of Psychology
University of Warwick
Coventry CV4 7AL
ENGLAND

Dr. Michael Atwood
Science Applications Institute
40 Denver Tech. Center West
7935 E. Prentice Avenue
Englewood, CO 80110

1 psychological research unit
Department of Defense (Army Office)
Campbell Park Offices
Canberra ACT 2600, AUSTRALIA

Dr. Alan Baddeley
Medical Research Council Applied
Psychology Unit
15 Chaucer Road
Cambridge CB2 2EF, ENGLAND

Dr. Patricia Baggett
Department of Psychology
University of Colorado
Boulder, CO 80309

Mr. Avron Barr
Department of Computer Science
Stanford University
Stanford, CA 94305

Cdr. Robert J. Biersner
Program Manager
Human Performance
Navy Medical R&D Command
Bethesda, MD 20014

Liaison Scientists
Office of Naval Research
Branch Office, London
Box 39 FPO New York 09510

Dr. Lyle Bourne
Department of Psychology
University of Colorado
Boulder, CO 80309

Dr. John S. Brown
Xerox Palo Alto Research Center
3333 Coyote Road
Palo Alto, CA 94304

Dr. Bruce Buchanan
Department of Computer Science
Stanford University
Stanford, CA 94305

Dr. Victor Bunderson
WICAT Inc.
University Plaza, Suite 10
1160 So. State St.
Orem, UT 84057

Dr. Pat Carpenter
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. John B. Carroll
Psychometric Lab
University of North Carolina
Davie Hall 013A
Chapel Hill, NC 27514

Charles Myers Library
Livingstone House
Livingstone Road
Stratford
London E15 2LJ, ENGLAND

Dr. William Chase
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Micheline Chi
Learning R&D Center
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. William Clancey
Department of Computer Science
Stanford University
Stanford, CA 94305

Dr. Allan M. Collins
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Lynn A. Cooper
LRDC
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. Meredith P. Crawford
American Psychological Association
1200 17th Street, N.W.
Washington, D.C. 20036

Dr. Hubert Dreyfus
Department of Philosophy
University of California
Berkeley, CA 94720

LCol. J.C. Eggenberger
Directorate of Personnel Applied Research
National Defense HQ
101 Colonel By Drive
Ottawa, CANADA K1A 0K2

Dr. Ed Feigenbaum
Department of Computer Science
Stanford University
Stanford, CA 94305

Dr. Richard L. Ferguson
The American College Testing Program
P.O. Box 168
Iowa City, IA 52240

Mr. Wallace Feurzeig
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Victor Fields
Department of Psychology
Montgomery College
Rockville, MD 20850

Dr. John R. Frederiksen
Bolt Beranek & Newman
50 Moulton Street
Cambridge, MA 02138

Dr. Alinda Friedman
Department of Psychology
University of Alberta
Edmonton, Alberta
Canada T6G 2E9

Dr. R. Edward Geiselman
Department of Psychology
University of California
Los Angeles, CA 90024

Dr. Robert Glaser
LRDC
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. Marvin D. Glock
217 Stone Hall
Cornell University
Ithaca, NY 14853

Dr. Daniel Gopher
Industrial & Management Engineering
Technion-Israel Inst. of Technology
Haifa, ISRAEL

Dr. James G. Greeno
LRDC
University of Pittsburgh
3939 O'Hara Street
Pittsburgh, PA 15213

Dr. Harold Hawkins
Department of Psychology
University of Oregon
Eugene, OR 97403

Dr. Barbara Hayes-Roth
The Rand Corporation
1700 Main Street
Santa Monica, CA 90406

Dr. Frederick Hayes-Roth
The Rand Corporation
1700 Main Street
Santa Monica, CA 90406

Dr. James R. Hoffman
Department of Psychology
University of Delaware
Newark, DE 19711

Glenda Greenwald, Ed.
"Human Intelligence Newsletter"
P.O. Box 1163
Birmingham, MI 48012

Dr. Earl Hunt
Department of Psychology
University of Washington
Seattle, WA 98105

Dr. Ed Hutchins
Navy Personnel R&D Center
San Diego, CA 92152

Dr. Steven W. Keele
Department of Psychology
University of Oregon
Eugene, OR 97403

Dr. Walter Kintsch
Department of Psychology
University of Colorado
Boulder, CO 80302

Dr. David Kieras
Department of Psychology
University of Arizona
Tuscon, AZ 85721

Dr. Kenneth A. Klivington
Program Officer
Alfred P. Sloan Foundation
630 Fifth Avenue
New York, NY 10111

Dr. Stephen Kosslyn
Harvard University
Department of Psychology
33 Kirkland Street
Cambridge, MA 02138

Mr. Marlin Kroger
1117 Via Goleta
Palos Verdes Estates, CA 90274

Dr. Jill Larkin
Department of Psychology
Carnegie-Mellon University
Pittsburgh, PA 15213

Dr. Alan Lesgold
Learning R&D Center
University of Pittsburgh
Pittsburgh, PA 15260

Dr. Michael Levine
Department of Educational Psychology
210 Education Bldg.
University of Illinois
Champaign, IL 61801

Dr. Charles Lewis
Faculteit Sociale Wetenschappen
Rijksuniversiteit Groningen
Oude Boteringestraat 23
9712GC Groningen
Netherlands

Dr. Erik McWilliams
Science Education Dev. and Research
National Science Foundation
Washington, D.C. 20550

Dr. Mark Miller
TI Computer Science Lab
C/O 2824 Winterplace Circle
Plano, TX 75075

Dr. Allen Munro
Behavioral Technology Laboratories
1845 Elena Avenue, Fourth Floor
Redondo Beach, CA 90277

Dr. Donald A. Norman
Dept. of Psychology C-009
Univ. of California, San Diego
La Jolla, CA 92093

Dr. Jesse Orlansky
Institute for Defense Analyses
400 Army Navy Drive
Arlington, VA 22202

Dr. Seymour A. Papert
Massachusetts Institute of Technology
Artificial Intelligence Lab
545 Technology Square
Cambridge, MA 02139

Dr. James A. Paulson
Portland State University
P.O. Box 751
Portland, OR 97207

Dr. James W. Pellegrino
University of California, Santa Barbara
Department of Psychology
Santa Barbara, CA 93106

Mr. Luigi Petrullo
2431 N. Edgewood Street
Arlington, VA 22207

Dr. Peter Polson
Department of Psychology
University of Colorado
Boulder, CO 80309

Dr. Steven E. Poltrock
Department of Psychology
University of Denver
Denver, CO 80208

Minrat M. L. Rauch
P II 4
Bundesministerium der Verteidigung
Postfach 1328
D-53 Bonn 1, GERMANY

Dr. Fred Reif
SESAME
c/o Physics Department
University of California
Berkeley, CA 94720

Dr. Andrew M. Rose
American Institutes for Research
1055 Thomas Jefferson St. NW
Washington, D.C. 20007

Dr. Ernst Z. Rothkopf
Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

Dr. Walter Schneider
Department of Psychology
University of Illinois
Champaign, IL 61820

Dr. Alan Schoenfeld
Department of Mathematics
Hamilton College
Clinton, NY 13323

Dr. Robert J. Seidel
Instructional Technology Group
HUMRRO
300 N. Washington St.
Alexandria, VA 22314

Committee on Cognitive Research
c/o Dr. Lonnie R. Sherrod
Social Science Research Council
605 Third Avenue
New York, NY 10016

Robert S. Siegler
Associate Professor
Carnegie-Mellon University
Department of Psychology
Schenley Park
Pittsburgh, PA 15213

Dr. Edward E. Smith
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

Dr. Robert Smith
Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

Dr. Richard Snow
School of Education
Stanford University
Stanford, CA 94305

Dr. Robert Sternberg
Department of Psychology
Yale University
Box 11A, Yale Station
New Haven, CT 06520

Dr. Albert Stevens
Bolt Beranek & Newman, Inc.
50 Moulton Street
Cambridge, MA 02138

David E. Stone, Ph.D.
Hazeltine Corporation
7680 Old Springhouse Road
McLean, VA 22102

Dr. Patrick Suppes
Institute for Mathematical Studies in
the Social Sciences
Stanford University
Stanford, CA 94305

Dr. Kikumi Tatsuoka
Computer Based Education Research Lab
252 Engineering Research Laboratory
University of Illinois
Urbana, IL 61801

Dr. John Thomas
IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Dr. Perry Thorndyke
The Rand Corporation
1700 Main Street
Santa Monica, CA 90406

Dr. Douglas Towne
University of Southern California
Behavioral Technology Labs
1845 S. Elena Avenue
Redondo Beach, CA 90277

Dr. J. Uhlaner
Perceptronics, Inc.
6271 Variel Avenue
Woodland Hills, CA 91364

Dr. Benton J. Underwood
Department of Psychology
Northwestern University
Evanston, IL 60201

Dr. Phyllis Weaver
Graduate School of Education
Harvard University
200 Larsen Hall, Appian Way
Cambridge, MA 02138

Dr. David J. Weiss
N660 Elliott Hall
University of Minnesota
75 E. River Road
Minneapolis, MN 55455

Dr. Gershon Weltman
Perceptronics Inc.
6271 Variel Avenue
Woodland Hills, CA 91367

Dr. Keith T. Wescourt
Information Sciences Department
The Rand Corporation
1700 Main Street
Santa Monica, California 90406